A Journey Through Computation

Natural and Human

Karl Fant

With gratitude to
Chris Nicol and Steve Johnson
for
listening and understanding

In memory of
Brian Cantwell Smith
who would have understood
if he could have listened

"For most of my life I have been unable to answer these questions, because I have not known what computation is." Brian Cantwell Smith*

"The whole story must be turned upside down."
Brian Cantwell Smith†

"It is we, not the physicists, who must develop a theory of everything."
Brian Cantwell Smith‡

*. Smith, Brian Cantwell. The foundations of computing. In Scheutz, Matthias (ed.), *Computationalism:New Directions:* Cambridge, MA: MIT Press, 2002. p 24

†. Smith, Brian Cantwell, *On the Origin of Objects*. Cambridge, MA: MIT Press, 1996. p x

‡. Smith, Brian Cantwell. The foundations of computing. In Scheutz, Matthias (ed.), *Computationalism:New Directions:* Cambridge, MA: MIT Press, 2002. p 53

Table of Contents

Introduction

> "Second, though, I also want to make evident just how much such a transformation costs. Politics, creativity, ambiguity, irreverence---none of these can be grafted at a later stage, onto a silent steel core, or even poured, like life-giving water, over inherently desiccated foundations. The whole story has to be turned upside down." Brian Cantwell Smith*

The journey of the upside down story commences with the discovery that the conventional primitives of computation, Boolean logic and stepwise sequentiality are not, in themselves, expressively primitive but require extrinsic expressional assistance. It is further realized that this extrinsic assistance is delivered by arbitrarily capable humans in the works who can make anything work including insufficiently expressive primitivities (Chapter 1).

In Chapter 2 a most primitive primitivity is proposed; a counterpoint of sameness and differentness with differentnesses spontaneously interacting and changing. Understanding that humans must be removed from consideration it is further proposed that the primitivities behave entirely on their own expressional merits in a universe containing only the behaving primtivities and nothing more. The journey exploring the evolution of computation within this universe discovers:

- an accounting of computation in which primitive behaviors compose to greater expressivity on their own intrinsic expressional merits in no need of extrinsic intervention and with no discontinuity in form of expression.
- a context of timeful computation in contrast to a context of timeless mathematics, in which time is a fully vested citizen of expression in contrast to an imposed mandate,
- a context of spontaneously, dependently and concurrently flowing wavefronts of transition in contrast to a context of explicitly controlled sequential steps of state update,
- a context in which determined sequence emerges from dependent concurrency in contrast to a context in which dependent concurrency emerges from determined sequence,
- a context providing a common characterization of disparate forms of computation whether in a biological cell, in a digital computer or in a mathematician's thoughts encompassing the real as well as the abstract, the biological as well as the logical, how nature computes as well as how humans compute,
- a context within which familiar forms of computation, Boolean logic with its time interval(section 5.7.5,) place value number with its arithmetic (see Dependency Language section 1.12) and sequential interpretation with its notions of sequence control and state update (section 4.11), emerge through considerations quite different from those of their historic development,

*. Smith, Brian Cantwell, *On the Origin of Objects*. Cambridge, MA: MIT Press, 1996. p x

Chapter 1:
Subtleties of primitivity

Sequence controlled steps of behavior, considered fundamental to computation, and Boolean logic, considered the most primitive of computational behaviors do not provide a complete and coherent accounting of computation.

1.1. Sequentiality.

> *"Computation is the evolution process of some environment by a sequence of "simple, local" steps."* *

Sequential computation is represented one step at a time. Each step is a behavior that receives an input from a state environment and returns an output to the state environment evolving the state environment step by step. Each step completes and delivers its output before the next step begins ensuring that each next step receives a stable input from a stable state environment.

1.1.1. The innate concurrency of sequentiality

A sequential ordering of steps does not, in itself, express a computation. A sequence of steps has to express the correct flow of dependency relations through the state environment. Dependency of flow relations include relations that can occur "all at once" or "in any order". The, "all at once", represents concurrent relations. The, "in any order", means that the concurrent relations can be mapped to a sequential ordering of steps. But "in any order" also means that there can be a multitude of different sequential orderings that correctly express the concurrent relations of the flow of dependency relations. Further, there is a multitude of sequential orders that do not correctly express the flow of dependency relations.

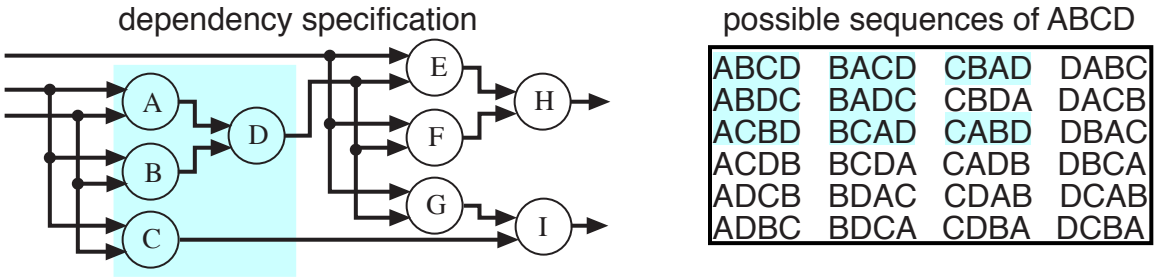


Figure 1.1. Sequencing concurrency

The left of Figure 1.1 represents a network of dependency relations. In the shaded region of the network **A** and **B** can occur in any order before **D**. If either **A** or **B** occur after **D** the sequence is incorrect. **C** can occur in any order with **A**, **B** and **D**. The right of Figure 1.1 represents all the possible sequences of **A**, **B**, **C** and **D**. The shaded sequences are the eight sequential orderings expressing the correct flow of dependency relations corresponding to the similarly shaded portion of the dependency network. The unshaded 16 possible sequences are all incorrect (**D** occurs before **A** or **B** or both). With such variety of sequence it can be difficult to be confident of the correctness of a specific sequential order and even more difficult to reliably perceive incorrectness. The only means of differentiating a correct sequential ordering from an incorrect

*. Avi Wigderson "Mathematics + Computation, The theory revolutionizing technology and science", (Princeton, New Jersey, Princeton University Press, 2019). p. 307.

sequential ordering in a typically enormous set of possible sequential orderings is to refer to the unique dependency of flow network expression with its innate concurrent relations.

1.2. QUANDARY 1: Sequential expression is not primitive

Sequential ordering necessarily derives from dependency relations with their innate concurrency. The expression of dependency relations with their innate concurrency must be considered to be more primitive than the expression of sequential order.

1.2.1. The sequence controller

Steps do not sequence themselves. Sequencing requires a sequence controller that actualizes each step in turn, connects it with the state environment and determines its completeness of output before beginning the next step. The mathematician with her pencil and paper is the original sequence controller. In the absence of the mathematician a sequence controller must be devised. A sequence controller might be devised as a sequence of more primitive behavior steps controlled by a more primitive sequence controller which in turn might be devised as a sequence of even more primitive behavior steps controlled by a even more primitive sequence controller forming a hierarchy of sequence controllers such as program, instructions, microcode and so on. However, there must be a most primitive sequence controller which cannot itself be expressed in terms of a controlled sequence of steps.

1.3. QUANDARY 2: Sequence control is not primitive

Sequence control which cannot express a most primitive sequence controller cannot be primitive. The most primitive sequence controller cannot itself be sequence controlled but must be expressed in terms of dependency relations among primitive behaviors which innately include concurrent dependency relations.

1.3.1. Concurrency

Concurrency relations are conventionally viewed from simply insignificant

> *“All we would lose by the omission of "parallel processing" is speed, nothing fundamental.” †*

to fundamentally unrealizable.

> *“The introduction of concurrency into computation opens Pandora’s box to release the possibility of non determinacy and a host of other complications, including deadlock and livelock .... Events within concurrent systems do not necessarily occur at predetermined times, but may occur in different orders depending upon the relative speeds of different system parts. The usual assumption, which is also physically realistic, is that relative speeds cannot be*

†. Richard Feynman, “*Feynman Lectures On Computation*” (Reading, Massachusetts, Addison Wesley, 1996) p 4.

controlled so precisely that the designer or programmer can depend upon them to enforce sequencing requirements." ‡

1.4. Boolean logic

Boolean logic behaviors, conventionally understood to be most primitive behaviors, are placed in a network of dependency relations with its innate concurrency relations. The enlivened Boolean behaviors are continually responsive to their input and are continually asserting output dependent on the input. Two behaviors, each presenting an input to a third behavior, can behave independently and concurrently with different delays. The two input transitions to the third behavior can arrive at different times causing the third behavior to temporarily output an incorrect result transition, a glitch, which will be presented to subsequent behaviors causing them to temporarily transition their output to incorrect result transitions. Glitches will race ahead though the network of behaviors asserting a chaos of incorrect transitions at the output of the network. The dependency network with its Boolean behaviors cannot, on its own, determine amidst the chaotic transitioning of incorrect results when the correct result of the presented input occurs. This chaotic transitioning through concurrent relations is the Pandora's box of non determinacy mentioned in the quote above.

1.4.1. Hiding the chaos of concurrency

If the inputs to a Boolean logic dependency network are held stably for long enough correct results eventually propagate through the behaviors and the network stabilizes asserting the correct output for the presented input. The chaotic behavior of a Boolean logic dependency network can be hidden and remediated by isolating and bounding it with memories controlled by a time interval long enough to allow the network to stabilize. At the beginning of the time interval an input memory presents the input to the network and an output memory ignores the output of the network. At the end of the time interval the output memory is enabled to accept the correct and stable output of the network. The chaotic concurrent behavior of the Boolean dependency network is isolated and hidden within the time interval and between the bounding memories. The deterministic expressivity of the Boolean dependency network is recovered but the dependency relations and their Boolean behaviors cannot compose beyond the time interval. From this point on composition is in terms of time intervals.

1.5. QUANDARY 3: Boolean logic is not primitive

A network of dependently related Boolean logic behaviors is not primitive. The Boolean logic behaviors, on their own behavioral merits, are insufficiently expressive to coordinate the innate concurrent relations of the dependency network and require the extrinsic expressional support of a time interval with memories.

1.5.1. Sequential steps of hidden concurrency

The Boolean dependency networks bounded by the clock are composed into a greater network of Boolean subnetworks all timed with the same time interval long enough to accommodate the slowest to stabilize subnetwork (critical path) behave concurrently (all at once) within each time interval (synchronously). The single common time interval precisely controlling

‡. Charles Seitz ed., *Resources in Parallel and Concurrent Systems*, ACM Press, New York, 1991, introduction, page ix.

the start and end of all the concurrently behaving subnetworks fulfills the above author's requirement of precisely controlled relative speeds for reliable concurrent behavior.

The control of the clock overcomes the problematic concurrency behaviors of the Boolean dependency network and enables the concurrent behavior of the composed subnetworks. In this way a most primitive sequence controller can be realized establishing sequential expression and sequence control.

1.6. QUANDARY 4: Confusions of primitivity

Sequence is not primitive. Sequence control is not primitive. A dependency network of Boolean logic behaviors is not primitive. Dependency relations while appearing to be primitive present chaotic concurrencies. The sequencing control of the clock overcomes the concurrency chaos allowing the realization of the most primitive sequence controller which enables sequential expression and interpretation. One could argue that sequence control is necessary and fundamental if not primitive. Yet the expression of sequenced steps at all levels still derives from and is preceded by dependency relations with their innate concurrency. Dependency relations with chaotic concurrency linger within each clock bounded Boolean logic network.

Dependency relations still underly everything.

1.6.1. Why Boolean logic?

Boolean logic is considered primitive to computation because it is considered mathematically primitive. In timeless mathematics the timing problems with concurrency do not arise. Furthermore, the arbitrarily expressive mathematician can realize a Boolean logic dependency expression step by step through time with her pencil and paper. She knows when new input data is present and when the output is done. She also knows to have the inputs to a concurrent relation resolved before resolving the concurrent relation. In particular she expresses temporal occurrence and temporal completeness in relation to the Boolean logic dependency network. Her time stepped Boolean network works just fine.

1.6.2. Boolean logic on its own

But a Boolean logic dependency network on its own without the mathematician's expression of temporal occurrence and temporal completeness relations cannot negotiate the concurrent relations of the dependency network. Boolean logic considered Mathematically fundamental can't be responsible for the chaotic behaviors. It must be the concurrency of the dependency relations. This unfortunate but unavoidable aspect of reality is an engineering problem, not a math problem.

1.6.3. The expressional crux of the primitivity matter

The Boolean behaviors are composed into a dependency network. The clock interval is imposed isolating the Boolean behaviors preventing them from further composing on their own terms. The unit of further composition becomes the time interval itself. There is a discontinuity in form of expressivity from composing dependency relations among Boolean behaviors to composing in terms of time steps. This incoherent discontinuity in form of expression does not support a coherent accounting of computation.

Sequentially stepping operations is indiscriminately expressive in that it does not intrinsically account dependency relations. There are a multitude of possible sequences of operations some correctly express a computation and a majority that compromise the dependencies and

incorrectly express the computation. There is nothing about sequencing that discriminates correct from incorrect. There must be an extrinsically imposed consideration of dependency relations with their intrinsic concurrency relations to realize a sequence of operations that correctly expresses a computation. This indiscriminate expressivity of sequence does not support a coherent accounting of computation.

Clocked Boolean logic and sequential programming works and has supported the phenomenal development of computing over the last 80 years. But still no one quite understands computation. This is because the computer is an artifact invented, built, programmed, supported, and used by an arbitrarily capable human agency. It is humans that construct a computer correctly adjusting the timing of the Boolean circuits. It is humans that program the computer ensuring correct sequences of operations. It is humans that build, maintain and use the computer.

If computation has a primitive essence it cannot be found in the artifact, in its sequential expression, sequence control, Boolean logic, in synchronicity or in its information state. It cannot even be found in the arbitrarily capably human making the Boolean logic work, providing the sequential expressions and information states.

As long as there are humans in the works rendering all possible primitivities workable there cannot be discovered an essential primitivity of computation. If there is an essence of computation it must precede the human and we must consider computation without humans.

> “Not only do we not understand computing as well as is generally thought, I will argue, but making progress will require upending all sorts of fundamental assumptions in ontology, epistemology, and even metaphysics." Brian Cantwell Smith[§]

§. Smith, Brian Cantwell. Brian Cantwell-Smith: From E&M to M&E: A journey through the landscape of computing. In Floridi, Luciano (ed.), *Philosophy of Computing and Information: 5 Questions*, Automatic Press, 2008, p. 19

Chapter 2:
Condition Differentiation

The pursuit of essential primitivity reaches beyond human centrality, mathematical formalism and even some laws of physics. The journey begins with a first principle of unstructured differentnesses spontaneously and dependently interacting and changing.[1] This is in contrast to regarding a mathematician with pencil and paper as first principle. The journey travels through a computational universe quite different from the universe with which you are conventionally familiar. It is a universe of relations of sameness and differentness, of persistence and change, of space and time, of one and many.

2.1. Sameness and differentness engender each other

The journey starts with the sameness of persistence and change of differentness. Change of differentness condition manifests only in relation to a referential sameness of persistence. Sameness of persistence of manifests only in relation to a referring change of differentness condition. Each individually signifies nothing but together -- a persistence asserting one at a time of two or more conditions of differentness -- they signify *a most primitive primitivity* that a universe can start with in the absence of theorizing humans to advise it.

2.1.1. Connect, constrain, extend

The sameness of each persistence and the changeability of its differentness conditions connect, constrain and extend each other.

2.1.1.1. Connect,

The oneness of a persistence connects its many different assertable conditions of differentnesses. The sameness of a persistence connects the **beforeness** and **afterness** of transition of its asserted differentness condition.

2.1.1.2. Constrain

The persistence constrains its many differentness conditions to **one at a time** assertion. Its differentness conditions constrain the persistence to asserting only its differentness conditions.

2.1.1.3. Extend

A persistence extends the expressivity of its differentness conditions beyond themselves by transitioning among them one at a time and by asserting each differentness condition multiple times. The differentness conditions extend the expressivity of the sameness of the persistence beyond itself with differentnesses of presentation.

2.1.2. Sameness and differentness of space and time

Each persistence is an exclusively singular indivisible same piece of space. No two persistences can be the same exclusively singular piece of space so each persistence is a different same piece of space.

Each persistence is also an exclusively singular immortal same piece of time. No two persistences can be the same exclusively singular piece of time so each persistence is a different same piece of time.

Each persistence is different from all other persistences but persistences are not intrinsically individuated and cannot be individually referenced by other persistences or by anything else.

Each persistence asserts one differentness condition at a time from a group of differentness conditions associated to the persistence. Differentness conditions are intrinsically individuated

and texture the differentness of space and time by lending their individuation to their asserting persistences and by their transitioning engendering unique differentnesses of time.

A persistence asserts its same differentness conditions over and over with each transition engendering and demarcating successive differentnesses of the persistence's immortal piece of time. Each differentness of persistence time is unique. There are no samenesses of persistence time that can occur over and over in persistence time.

A persistence with its asserted condition of differentness manifests an individuated differentness of condition space. A persistence transitioning its condition of differentness manifests a unique differentness of persistence time.

Each persistence is both sameness and differentness **of** both space and time.
Each differentness condition is a differentness **in** samenesses of both space and time.

This is not the space and time with which you are familiar.

2.1.3. Illustration

Persistences and their differentness conditions are illustrated with circles, colors and arrows. In Figure 2.1 a circle represents a persistence. Color represents its asserted differentness condition. The connected arrows represent a same persistence transitioning differentness conditions. Each transition represents a unique differentness of persistence time, **Tx**.

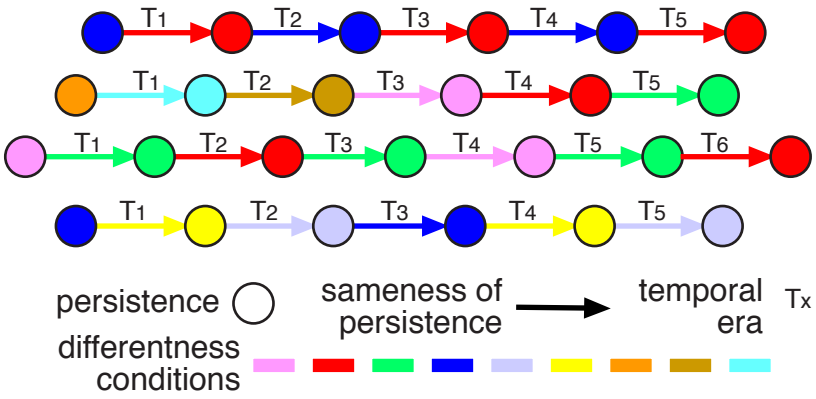


Figure 2.1. Persistences and their transitioning differentness conditions.

2.2. Interaction: Differentness apreciating differentness

If the asserted differentness condition of a persistence spontaneously transition then the changing of the differentness conditions and the sameness of the persistence are mutually relating but the whole is not relating and does not extend beyond itself. Persistences and their asserted differentness conditions can extend beyond themselves if each asserted differentness condition transitions only in dependent interaction with another differentness condition asserted by a different persistence.

2.2.1. Propensity to specifically interact

Differentness conditions are individuated and individually reference each other with interaction propensities. Each differentnesses condition has a propensity to transition only in interaction with specific other differentness conditions and to not transition, not interact, with all other differentness conditions.

2.2.1.1. Nature's persistences and their differentness conditions

Having introduced the abstract notion of differentness with constrained propensity to interact we note that nature is replete with such entities. A chemical molecule is a persistence that asserts a chemical differentness condition that may spontaneously interact with and change its chemical condition only in interaction with specific other chemical conditions while the molecule persistently remains a molecule relating the before and after chemical differentness conditions. One molecule asserting differentness condition sodium interacts with one molecule asserting differentness condition chlorine to form one molecule of differentness condition salt. 92 elements, chemical molecules and in particular biochemical molecules, proteins, RNA, DNA provide a near infinite supply of differentness conditions in the form of proteins each with propensity to interact only with specific other differentness conditions. Computation begins with these entities. Where they might have come from and why they might have specific interaction propensities is a different matter.

This journey is only about computation which commences with the advent of atoms and molecules with specific interaction propensities. Persistences and their assertable differentness conditions with their interaction propensities constitute the received, unsolicited ontology through which the journey travels.

2.2.2. Interaction

Persistences wander freely through condition space. When two or more persistences encounter as in Figure 2.2 their asserted differentness conditions either interact or do not interact. If the asserted differentness conditions of the encountering persistences have a propensity to interact they interact with their interacting differentness conditions transitioning to result differentness conditions then the persistences un-encounter as with **interaction** of Figure 2.2. **Red** differentness condition and **green** differentness condition have amenable propensity to interact. A persistence asserting differentness condition **red** and a persistence asserting differentness condition **green** encounter and interact with the **red** differentness condition transitioning to **blue** and the **green** differentness condition transitioning to **brown** after which the persistences un-encounter. If the encountering differentness conditions have no propensity to interact, as **no interaction** of Figure 2.2 with **orange** and **green** then nothing happens. Each persistence with its asserted differentness condition un-encounter with no lingering consequences of the failure to interact. Other interactions include the **persistence union interaction** in which two or more persistences combine to form a single persistence asserting a single differentness conditions. The **persistence disunion interaction** in which one persistence uncombines to form a two or more persistences each asserting a differentness conditions. The **catalytic interaction** in which one persistence **red** interacts with **green** but **green** does not interact with **red**.

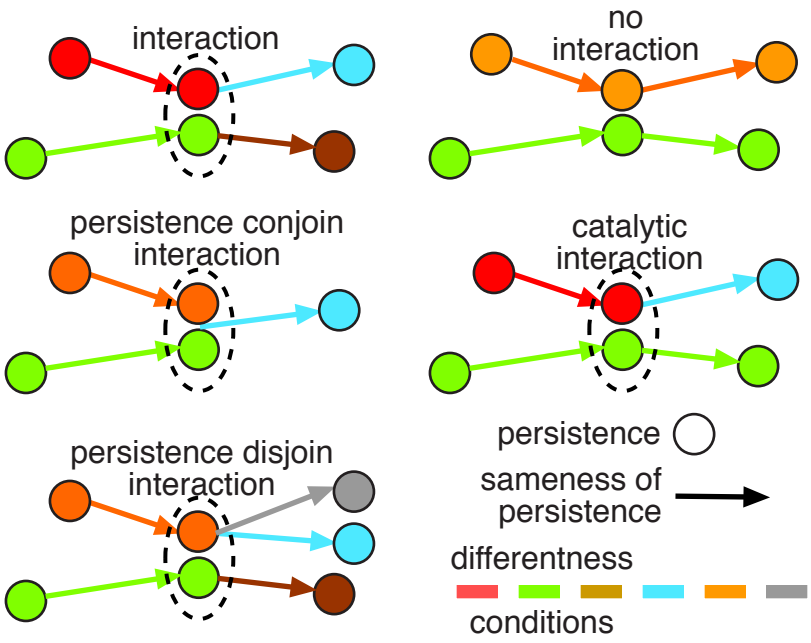


Figure 2.2. Encounters of differentness conditions.

Interaction is differentness appreciating differentness through a sameness of behavior. Differentness conditions that encounter but do not interact are computationally immaterial.

2.2.3. Directionality

Interaction is directional in the sense that the result differentness conditions do not re-interact and transition back to the interacting differentness conditions. If the result differentness conditions re-transition in relation to each other, i.e., if in Figure 2.2 **blue** re-transitions to **red** and brown re-transitions to **green**, then the behavior isolates itself and does not extend the expressivity of the persistences and their asserted differentness conditions beyond themselves, i.e., there is no effective interaction.

2.2.4. Completeness criterion

An interaction occurs only when all of the interacting differentness conditions are sufficiently proximally encountered. An interaction is complete when the interacting differentness conditions disappear and the interaction result differentness conditions appear as with the **interaction** of Figure 2.2 in which the **red** and **green** differentness conditions disappear and the **blue** and **brown** result differentness conditions appear The appearance of interaction result differentness conditions indicates that the interacting differentness conditions were completely present, sufficiently proximally encountered, and that the interaction has completed.

2.2.5. Egalitarian interaction

The interacting differentnesses observe and register each other equally. Each is both observing subject and observed object. Neither participant causes the interaction. No interaction participant is privileged over any other interaction participant. Interaction itself is not privileged. It just occurs incidentally and spontaneously. There is no meta. There is only mutuality.

2.2.6. Mutual dependence

Interaction is dependent on the encounter of different persistences asserting differentness conditions amenable to interact. The persistences with their asserted differentness conditions are dependent on the interaction to extend their expressivity.

2.2.7. Minimal expressivity

One might suggest that a persistences and its asserted differentness condition can be observed simply by causing a particular kind of interaction but there is no agency capable of intentionally referencing and manipulating a persistence with its asserted differentness condition to cause a particular kind of interaction.

One might suggest that the differentnesses and their interaction propensities can be designed, and their behavior programmed. But at this stage of the journey there is no designer, no builder, no programmer, no intentionality, no causality.

There is only the persistences and their asserted differentness conditions incidentally and spontaneously interacting and transitioning, nothing more. They are entirely on their own with nothing to tell them how to be or what to do.

2.2.8. Connect, constrain, extend

The sameness and oneness of interaction connects two or more different persistences asserting differentness conditions with propensities to interact and connects the interacting differentness conditions to result differentness conditions.

The interaction propensities of the differentness conditions constrain the occurrence of interaction and the appearance of the result differentness conditions.

Interaction extends the expressivity of differentness conditions and their asserting persitences beyond themselves.

2.2.9. Condition space

A multitude of persistences asserting differentness conditions freely encounter, un-encounter and re-encounter within the emptiness of a reflectively bounded condition space, like molecules in a drop of liquid bounded by surface tension. A bounded space, which is itself a spanning persistence of sameness, might be a gravity well, a planetary ocean within a gravity well, a tide pool within an ocean, a bubble within a tide pool, a cell membrane, a concentration gradient, a community, and so on. The reflective boundary encloses the space ensuring that persistences continually encounter rather than wandering off and not encountering. There is no intrinsic structure to the boundary other than it encloses the space and is reflective to the encompassed persistences.

It is a space of differentness conditions: condition space.

2.2.9.1. No intrinsic metric of space

One might expect that each persistence must occupy an individually registrable place in condition space but there is nothing capable of registering any relative orientation in condition space. There is nothing capable of presuming an inertial reference frame, of projecting a dimensional metric in relation to the reference frame or of registering measurement in relation to the projected metric, i.e., there are no humans. There is no intrinsic property of relative placeness or relative orientation in condition space and there is nothing capable of imposing such.

To continue on this journey you have to set aside the notion of three dimensional space and continuous time as well as the notion of any distal observer.[2] There are only the persistences

asserting differentness conditions encountering and unencountering in condition space. Nothing more.

2.2.9.2. Intrinsic dimensionality

The intrinsic dimensionality of condition space lies in the orthogonal properties of the free flowing persistences and their asserted differentness condition with their interaction propensities. The differentness conditions, indifferent to their asserted persistences, and the persistences, indifferent to their assertable differentness conditions, form orthogonal axes of condition space. The axes intersect when indiscriminately diffusing persistences encounter and their asserted differentness conditions interact. The dimensional axes are dynamic and self registering. The space observes itself.

2.2.9.3. Interactive encounter

The only registerable spatial relation is persistences encountering and their asserted differentness conditions interacting and transitioning. The interaction itself is the registration of the encounter. Non interactive encounters are not registerable. Persistences asserting differentness conditions interactively encounter or not. Nothing more can be said.

No other spatial orientation is registrable either relative to other persistences including non-interactive encounters or relative to the space itself including encounters with the reflective boundary. Of such, nothing can be said. Even when humans arrive with their theories and experiments they will not be able to speak of anything beyond direct interactive encounter.

2.2.9.4. Simultaneity and ephemerality

Interaction is ephemeral. The persistences encounter. The interacting differentness conditions disappear. The result differentness conditions appear. And the persistences un-encounter. The only thing left of the interaction is each result differentness conditions and their transitioned differentnesses of persistence time all of which simultaneously transitioned in the interaction.

2.2.9.5. Concurrency

There are countless persistences indiscriminately encountering in condition space with a multitude of them interacting.

2.2.10. Persistence time

An interaction engenders unique differentnesses of persistence time with its transitioning differentness conditions. Each persistence and its transitioning differentness conditions establish its own immanent manifestation of time with the **“one at a timeness”** of differentness condition assertion and with the **“beforeness and afterness”** of differentness condition transition. The **“beforeness and afterness”** of the transitioning of the asserted differentness conditions of a persistence is, for the persistence, a tick of time, of transitioning between unique differentnesses (eras) of persistence time. The **“one at a timeness”** of the assertion of a differentness condition by a persistence pursuing its next interaction between interaction transition ticks is, for the persistence, a tock of time: a uniquely different successive era of the persistence’s same immortal piece of time.

2.2.10.1. Dependency of time

The transitioning of differentness of time follows the transitioning of asserted differentness conditions rather than the transitioning of differentness conditions following the transitioning of differentness of time (a circuit clock for example). Interaction transition engenders differentness of time. Differentness of time does not engender interaction transition.

2.2.10.2. Ephemerality of time

When a differentness condition disappears by transitioning in an interaction its differentness of persistence time disappears with it. While a disappeared differentnesses of condition can be reasserted by a persistence, a disappeared differentness of persistence time is irretrievably lost and cannot recur. There are no samenesses of persistence time. There are only unique differentnesses of persistence time, a succession of afters. When a differentness of persistence time transitions to an after nothing remains of the before. The persistence retains no memory of the before and there can be no references to before.

2.2.10.3. No intrinsic metric of time

There is no intrinsic metric of global time. The notion of an intrinsic nature of time flowing of itself equably without regard to anything external relative to the universe itself and everything in it is not a part of this journey.

One might insist that as the persistences are wandering around in condition space from here to there, then to there, then to there and so on there must be passage of time but there are no theres of space (see 2.2.9.1) and there are no overarching thens of time. There is nothing capable of registering such differentnesses of time and space and there are no referents for such registrations. Such registration is not necessary to the universe neither is it useable to the denizens of condition space.

2.2.10.4. The ultimate extender

An asserting persistence extends the expressivity of its differentness conditions by asserting the same differentness conditions over and over through differentnesses of persistence time. Persistence time, intrinsically self extending, is the inexhaustibly different extender of sameness. There are no same differentnesses of persistence time to be extended by recurring over and over through differentness of persistence time.

2.2.10.5. Space and time

The only registration of space and time at this point is the interaction registering the encounter of persistences in condition space, the transition of their differentness conditions and instants of transition of persistence times.

2.3. The wavefront: interaction appreciating interaction

While there can be a multitude of interactions occurring independently a single interaction does not extend beyond itself. The expressivity of interaction extends beyond itself with a successor interaction dependent on the result differentness conditions of predecessor interactions. Interaction amenable differentness conditions encounter, interact and disappear transitioning into result differentness conditions which in their turn encounter amenable differentness conditions, interact and disappear transitioning into result differentness conditions which in their turn encounter amenable differentness conditions and so on. With each interaction the interacting differentness conditions disappear and result differentness conditions appear flowing as a wavefront from interaction to interaction.

2.3.1. A network of dependency relations

A progression of interactions each dependent on the result differentness conditions of predecessor interactions forms a network of dependently connected interactions. The wavefront of result differentness conditions, flowing through the dependency network from interaction to interaction embodies this network. The wavefront contains in its result differentness conditions

the accumulating result of the network of dependent interactions extending the results of each interaction to a result of the network of interactions.

2.3.1.1. Interaction ephemerality

Each individual interaction with its interacting differentness conditions disappears and its result differentness conditions in their turn will disappear with their next interaction. This ephemerality of interaction is transcended by the influence of its result differentness conditions on subsequent interactions through a network of dependency relations traversed by a wavefront carrying the influences from interaction to interaction.

2.3.1.2. The ephemeral network and its wavefront

Figure 2.3 illustrates the interaction by interaction flow of a wavefront through a network of dependent interactions shown in frame **0**. The wavefront starts in frame **1** with five persistences asserting differentness conditions seeking their next interaction and proceeds frame by frame through the network of dependent interactions with result differentness conditions appearing and interaction differentness conditions and their interactions disappearing. Frame **7** is the same five persistences asserting result differentness conditions looking for their next interaction. The appearance of result differentness conditions forms the leading edge of a wavefront and the disappearance of the interacting differentness conditions forms the trailing edge of a wavefront.

The dependency network is substantively represented by the flowing wavefront.

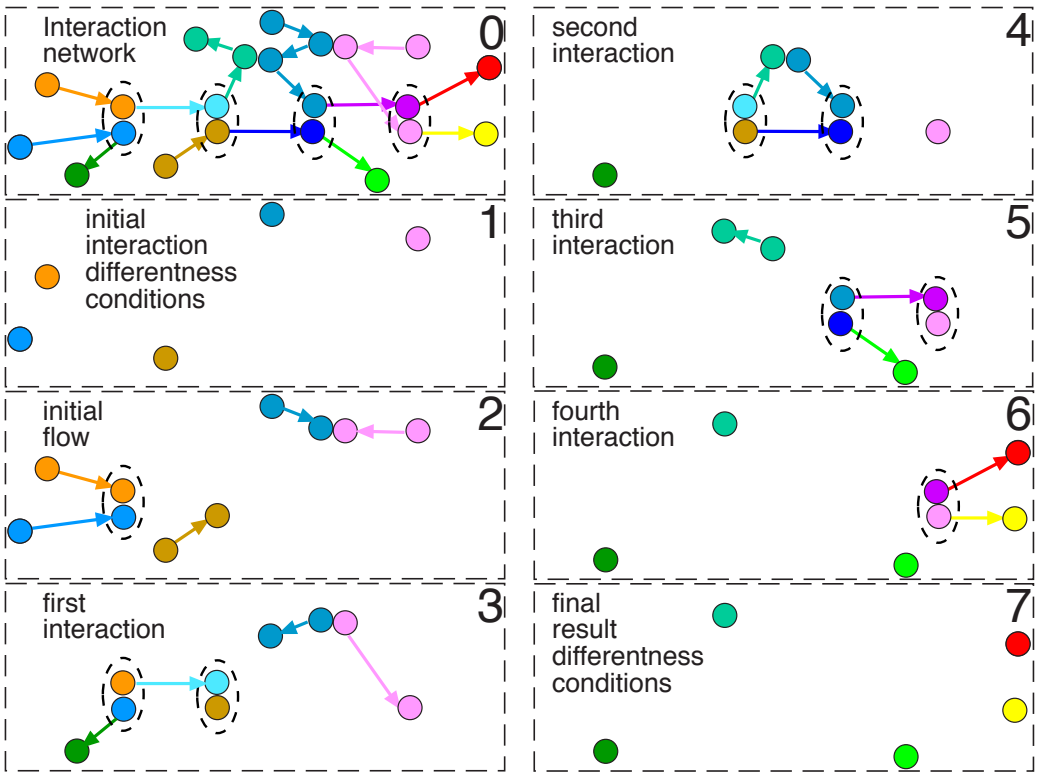


Figure 2.3. Wavefront flow through a network of dependent interactions.

2.3.1.3. Hand off propagation

In an interaction all of the persistences transition their differentness conditions and any of the result differentness condition can continue a wavefront. This can be seen in Figure 2.3 with a different persistence continuing the wavefront flow from each interaction. The persistences

asserting the **orange** and **green** initially interacting differentness conditions did not flow through all of the interactions to assert the final **red** and **yellow** result differentness conditions.

While a persistences extends beyond itself with its asserted differentness conditions this extension is limited by its complement of differentness conditions. With handoff propagation a wavefront is not limited by the differentness condition complement of any individual persistences but further extends their expressivity through condition space and differentness of time.

2.3.1.4. Concurrency in the network

An interaction wavefront flowing through the dependency network can include concurrent interactions coordinated by the dependency relations among the differentness conditions as shown in Figure 2.4. In the dependency network interactions **B** and **C** occur after interaction **A** and before interaction **D**. Nothing more can be said. In particular nothing can be said about any order between **B** and **C** as well as about both at once. One might suggest that, in principle, there must be a temporal relation between **B** and **C** but there is nothing that can appreciate such a temporal relation between **B** and **C**. The principle is moot.

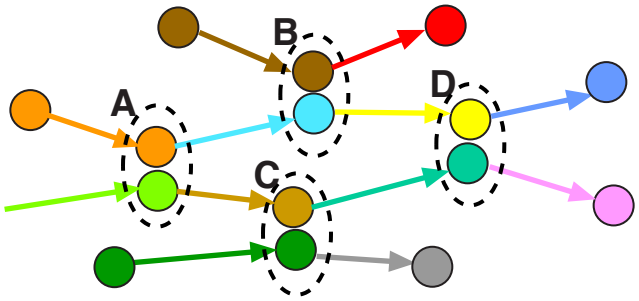


Figure 2.4. Concurrent interaction in a dependency network.

2.3.2. Substantiveness

Substantiveness is construed as a network of coherently behaving dependency relations extending through condition space and through differentness of condition time. The interaction wavefront is the first substantive dependency relation with the possibility of extending indefinitely through space and time. If a result differentness condition does not continue on to further interaction the result differentness condition no longer speaks for itself, nothing else speaks for it, it conveys no influence and nothing more can be said of it.

A wavefront does not intentionally strive to continue but if a wavefront ever does achieve continual propagation it is significant and substantive. This journey is the story of the evolution of substantiveness, of increasing coherent extension through dependency relations in condition space and through differentnesses of persistence time.

2.3.3. Connect, constrain, extend

The Each interacting persistence connects interaction differentness conditions to result differentness conditions. The result differentness conditions connects each interaction to successor interactions. The wavefront flowing with its result differentness conditions from interaction to interaction connects the interactions as a dependency network.

The wavefront of result differentness conditions is constrained by the interaction propensities of differentness conditions.

The wavefront carrying the accumulating result of the progression of dependent interactions extends the expressivity of the interactions possibly indefinitely.

2.3.4. Wavefront space

Persistences seamlessly connect interaction differentness conditions through an interaction to result differentness conditions and then deliver result differentness conditions to next interactions which seamlessly connect them as interacting differentness conditions to result differentness conditions and so on forming a wavefront of result differentness conditions seamlessly flowing from interaction to interaction. The seamlessly flowing wavefront forms a composite exclusive singularity of condition space inherited from the permanent exclusive singularity of its component persistences, an ontological object not because it was anointed by a referencing distal subject but because it is a coherent substantive sameness extending through dependency relations in condition space and through differentnesses of persistence time.

2.3.5. Wavefront time

Each result differentness condition also represents a unique differentness of persistence time in relation to its asserting persistence. The common instant of an interaction synchronizes the individual differentnesses of persistence time represented by the result differentness conditions of the interacting persistences. Each persistence asserting a result differentness condition with its differentness of persistence time flows to a next interaction whose synchronizing instant of time engenders the next instance of time for each of its interacting persistences. Each successive dependent interaction engenders successive afternesses of persistence time. The wavefront, carrying the result differentness conditions from interaction to interaction, is also carrying synchronized afternesses of persistence times extending each individual persistence's differentnesses of time into a coherent flow of successive dependent afters in relation to the wavefront. The wavefront, itself, becomes a coherent flow of time extended beyond the individual persistences.

Without interactions there is no differentness of time. Without a wavefront flowing from interaction to interaction there is no flow of differentness of time.

2.3.5.1. Bootstrapping substantiveness

Above (section 2.3.2), we spoke of the wavefront becoming substantive with its flowing through differentnesses of time, but the wavefront itself is establishing the flow of differentness of time through which it is flowing. It is bootstrapping its own substantiveness. There is no intentionality here. It is purely circumstantial. It just so happens that the wavefront establishes its own substantiveness with its own behavior.

2.3.5.2. Insubstantive time

Time is not directly expressed but is indirectly represented by asserted differentness conditions, each of which represents a differentness of time in relation to its asserting persistence. Differentnesses of time do not interact or register. Time has no substantiveness itself. The denizens of the environment go about their business in terms of dependency relations and chaotic encounter with no concern for time. Nothing so far references time in any sense. Interaction behavior establishes a flow of time which nothing registers and which has no influence.

2.3.5.3. But we humans need to appreciate time?

But we humans really want to understand what is going on in terms of the evolution of substantiveness and that requires an appreciation of time because differentness of time is referentially in the middle of everything as an arbiter of substantiveness. We cannot just project our notion of a metric time onto the story because we do not exist in the story.

In the computational universe time is not a metric. It is a dependent flow of local afters.

The computational universe does not have any metrics.

2.3.5.4. Primitivity of time

Time, to have any relevance, to even exist, must emerge from primitivity. To this end, the story characterizes time as emerging from the individual immortalities of the multitude of persistences, each persistence partitioned into a succession of temporal afters by the transitions of its asserted differentness conditions. Individual persistence afters are synchronized by the sameness of interaction and progressively extended by the wavefront of result differentness conditions representing successive differentnesses of time flowing from interaction to interaction engendering the wavefront flow of time.

2.3.5.5. The wavefront memory

Result differentness conditions are constant from interaction to interaction as is their differentness of persistence time. One might say that a wavefront remembers the result differentness conditions and their differentnesses of time between each interaction. But *remember* is one of those human words the usage of which implies more than is necessary. Remember is not an appropriate word here since as far as the wavefront is concerned there is no passage of time through which to remember. Nevertheless, we have to use our words.

One dilemma of this journey is at what point can we credibly apply computation relevant human words such as compute, effective, mechanism, machine, information, intentionality, sequence, state, syntax, semantics and so forth to the landscape of the journey.

2.3.6. Emergent substantiveness

The wavefront is an emergent substantiveness. There is nothing that registers, references or observes the wavefront. One might suggest that the interactions register the wavefront but they disappear. It is the wavefront, not disappearing, that registers the ephemeral interactions. It might be the persistences registering the wavefront but, even though the persistences do not disappear, they trade off as the wavefront flows and no persistence can register the wavefront as an extended sameness. It is the wavefront that registers the dependent flow of persistences and their asserted differentness conditions from interaction to interaction.

Each wavefront unappreciated, unreferencable, unobservable is entirely on its own and will survive through condition space and differentness of persistence times or it will not.

2.3.7. A weave of wavefronts

With a flowing wavefront there are differentness conditions flowing peripherally into the wavefront interactions that are not integral to the coherence of the wavefront flow. To this point the provenance of these peripheral differentness conditions has not been considered. However they can be differentness conditions of other wavefronts. Wavefronts flowing through wavefronts weaving a tapestry of dependently interconnecting wavefronts leading to complex networks of dependent wavefronts.

2.4. A familiar example of pure condition differentiation: Roman numeral addition

To illustrate that we are indeed talking about computation a familiar computation, Roman numeral addition, is presented with pure condition differentiation. The Roman numerals are expressed without the subtractive principle: 9 is **VIIII** instead of **IX**, 40 is **XXXX** instead of **XL** and so on. The subtractive principle which may have been invented by stone carvers as an

economy measure is not intrinsic to Roman numerals.* A Roman numeral is represented by the **possible differentness conditions**: **I**, **V**, **X**, **C**, **D**, **M**. Without the subtractive principle Roman numerals are association independent. Place in relation does not contribute to differentiation.

XLII = LXII = ILXI = IILX = LIIX = LIXI = IXIL = XILI = XIIL = IIXL = … = 62

The bounded condition space of indiscriminate encountering is represented by a shaking bag, illustrated in Figure 2.5. The bag prevents persistences and their asserted conditions from wandering off and prevents external persistences from intruding. The shaking conveniently illustrates and emphasizes the independent and un-oriented relations of the persistences within the space which ensures that each asserted differentness condition encounters all the other asserted differentness conditions in the bag and that all possible interactions occur.

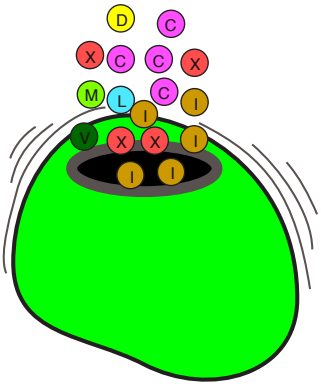


Figure 2.5. Persistences indiscriminately associate in the shaking bag..

Roman numeral addition is expressed with the interaction propensities shown in Figure 2.6.

[I,I,I,I,I] => V
[V,V] => X
[X,X,X,X,X] => L
[L,L] => C
[C,C,C,C,C] => D
[D,D] => M

Figure 2.6. Roman Numeral interaction propensities.

The differentness conditions embody their interaction propensities. When two **V**s encounter they fulfill the interaction relation [**V**,**V**] and spontaneously change into **X**. Given two Roman numerals these interaction propensities will reduce them to a minimal single numeral representing their addition. The 1896 + 341 = 2237 is the example addition.

MDCCCLXXXXVI + CCCXXXXI = MMCCXXXVII

The two numbers to be added are placed into a shaking bag as in Figure 2.7 below. The five **X**s that fulfill the relation [**X,X,X,X,X**], and change into a **L**. The five **C**s that fulfill the relation

*. See “https://en.wikipedia.org/wiki/Roman_numerals”.

[**C**,**C**,**C**,**C**,**C**], and change into a **D**. There are then two **L**s that fulfill the relation [**L**,**L**], and change into a **C** and two **D**s that fulfill the relation [**D**,**D**], and change into a **M.** The remaining differentness conditions do not fulfill any interaction propensity and are the answer to the addition.

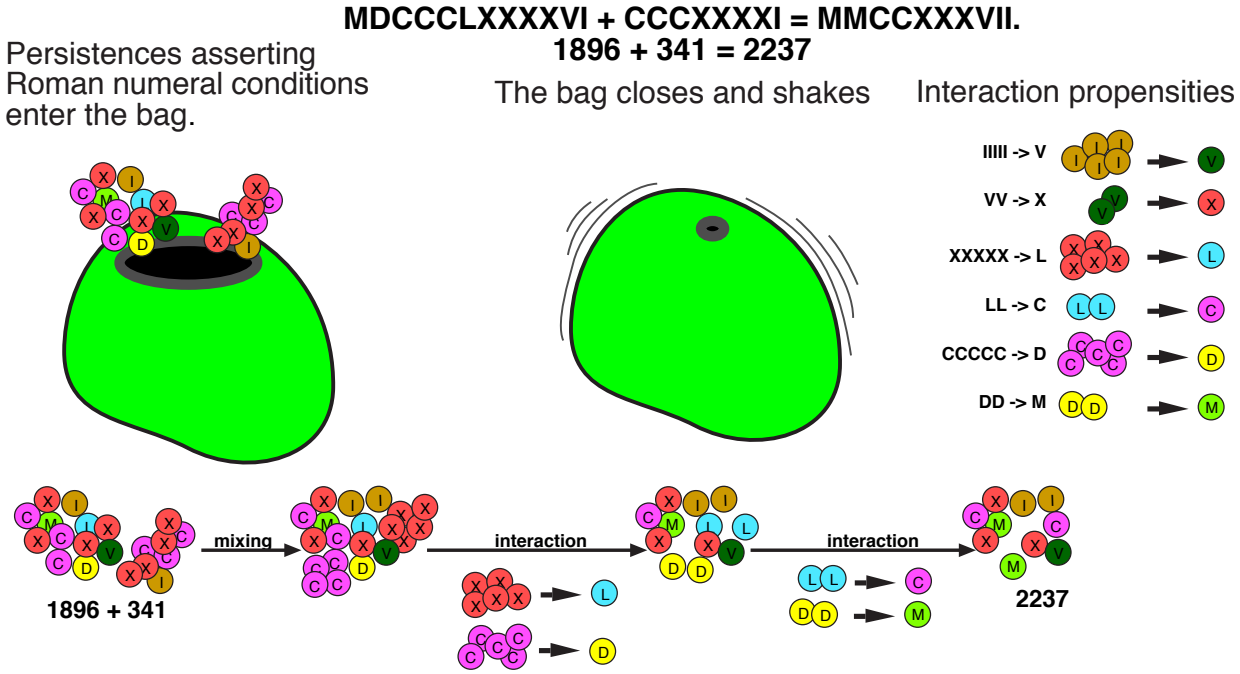


Figure 2.7. Roman numeral addition in shaking bag.

A fully determined numerical interaction occurs as a progression of discriminate interactions of differentness conditions according to interaction propensities while differentness conditions indeterminately associate inside the shaking bag

2.4.1. Interaction incompleteness

But the addition interaction is not complete in itself. It cannot, itself, determine when its interaction is completed. As with a Boolean logic network the only way to determine the progress of the addition is with an external agency which, in this case, must open the bag and perform a complicated inventory of the differentness conditions. When there are four or fewer of **I**, **X**, **C** and one or fewer of **V**, **L**, **D** the addition is completed.

2.4.2. Completeness of expression

Intrinsically determining when an interaction is done requires that there be a necessarily last interaction propensity fulfilled. With the present form of the expression there might not even be a first interaction. **VI** + **XII** = **XVIII** is done with no interaction behavior at all. There must be a completeness of behavior at each stage of interaction to insure that every interaction propensity is fulfilled in an orderly progression to the necessarily last interaction.

Completeness of behavior requires a completeness of representation. The first question of an addition is, how many **I**s are present in the bag. The interaction must count the **I**s and somehow determine that it has considered all the **I**s and has not missed any **I**s in the bag. This counting can be accomplished if the number of **I**s in the bag to be counted is always the same. This constancy of quantity of Is is arranged with buffer conditions such that there is always the same number of each condition and its buffer condition in each Roman numeral. The corresponding lower case

letter will be used as the buffer condition for each numeral condition as shown in Figure 2.8.

Numeral condition	Buffer condition
I	**i**
V	**v**
X	**x**
L	**l**
C	**c**
D	**d**
M	**m**

Figure 2.8. Roman numerals with their associated buffer conditions.

The buffer condition **i** is used such that there is always exactly four of **I** and/or **i** in a Roman numeral: **iiii**, **Iiii**, **IIii**, **IIIi** or **IIII**. The buffer conditions serve a role similar to zero in place value numbers. When two numerals are added there will always be exactly eight of **I** and/or **i**. The criterion for completeness, the counting of **I/i,** can now be expressed by the completeness fulfillment of each **I/i** Interaction propensity, shown in Figure 2.9 below, which is exactly eight **I/i** conditions. Each completeness of association fulfillment is an unordered association of conditions. There are no order relationships in the bag. The Interaction propensity fulfillment will occur only when the proximal association of all eight conditions is completely formed.

Interaction propensities `for I,i`

```
[i,i,i,i,i,i,i,i] => [i,i,i,i,v]
[I,i,i,i,i,i,i,i] => [I,i,i,i,v]
[I,I,i,i,i,i,i,i] => [I,I,i,i,v]
[I,I,I,i,i,i,i,i] => [I,I,I,i,v]
[I,I,I,I,i,i,i,i] => [I,I,I,I,v]
[I,I,I,I,I,i,i,i] => [i,i,i,i,V]
[I,I,I,I,I,I,i,i] => [I,i,i,i,V]
[I,I,I,I,I,I,I,i] => [I,I,i,i,V]
[I,I,I,I,I,I,I,I] => [I,I,I,i,V]
```

Interaction propensities `for V,v`

```
[v,v,v] => [v,x]
[V,v,v] => [V,x]
[V,V,v] => [v,X]
[V,V,V] => [V,X]
```

Figure 2.9. Interaction propensities for I,i and V,v.

The result of the interaction of eight **I/i** is four **I/i** and one **V/v**. The one **V/v** is the carry to the **V/v** interaction. There is one of **V/v** in each Roman numeral and there will always be a carry condition of **V** or **v** so there will always be exactly three of **V/v** present after the carry. The **Vv** Interaction propensity fulfillments require three of **V/v** in Figure 2.9 above ensuring that the **V/v** interaction occurs strictly after the **I/i** interaction. The **V/v** interaction produces one of **V/v** and one of **X/x**.

There will be four of **X/x** in each Roman numeral so adding two Roman numerals will involve eight **X/x** conditions and the carry **X/x** condition making exactly nine**X/x** conditions to form the **X/x** Interaction propensity fulfillments shown in Figure 2.10. Again, because of the carry, the **X/x** interaction will occur strictly after the **V/v** interaction.

Interaction propensities `for X,x`

```
[x,x,x,x,x,x,x,x,x] => [x,x,x,x,l]
[X,x,x,x,x,x,x,x,x] => [X,x,x,x,l]
[X,X,x,x,x,x,x,x,x] => [X,X,x,x,l]
[X,X,X,x,x,x,x,x,x] => [X,X,X,x,l]
[X,X,X,X,x,x,x,x,x] => [X,X,X,X,l]
[X,X,X,X,X,x,x,x,x] => [x,x,x,x,L]
[X,X,X,X,X,X,x,x,x] => [X,x,x,x,L]
[X,X,X,X,X,X,X,x,x] => [X,X,x,x,L]
[X,X,X,X,X,X,X,X,x] => [X,X,X,x,L]
[X,X,X,X,X,X,X,X,X] => [X,X,X,X,L]
```

Interaction propensities `for L,l`

```
[l,l,l] => [l,c]
[L,l,l] => [L,c]
[L,L,l] => [l,C]
[L,L,L] => [L,C]
```

Figure 2.10. Interaction propensities for X,x and L,l.

After the **X**/**x** interaction there will be three **L**/**l** conditions fulfilling the **L**/**l** Interaction propensities strictly after the **X**/**x** interaction. The **L**/**l** interaction produces one **L**/**l** condition and one **C**/**c** carry condition.

There will be four of **C**/**c** in each Roman numeral so adding two Roman numerals will involve eight conditions and the carry condition will make exactly nine **C**/**c** conditions to form the **C**/**c** interaction propensity fulfillments as shown in Figure 2.11. Again, the **C**/**c** interaction will occur strictly after the **L**/**l** interaction.

Interaction propensities for **C,c**

```
[c,c,c,c,c,c,c,c,c] => [c,c,c,c,d]
[C,c,c,c,c,c,c,c,c] => [C,c,c,c,d]
[C,C,c,c,c,c,c,c,c] => [C,C,c,c,d]
[C,C,C,c,c,c,c,c,c] => [C,C,C,c,d]
[C,C,C,C,c,c,c,c,c] => [C,C,C,C,d]
[C,C,C,C,C,c,c,c,c] => [c,c,c,c,D]
[C,C,C,C,C,C,c,c,c] => [C,c,c,c,D]
[C,C,C,C,C,C,C,c,c] => [C,C,c,c,D]
[C,C,C,C,C,C,C,C,c] => [C,C,C,c,D]
[C,C,C,C,C,C,C,C,C] => [C,C,C,C,D]
```

Interaction propensities for **D,d**

```
[d,d,d] => [d,m]
[D,d,d] => [D,m]
[D,D,d] => [d,M]
[D,D,D] => [D,M]
```

Figure 2.11. Interaction propensities for C,c and D,d.

After the **C**/**c** interaction there will be three **D**/**d** conditions fulfilling the **D**/**d** Interaction propensities strictly after the **C**/**c** interaction. The **D**/**d** interaction produces one **D**/**d** condition and one **M**/**m** carry condition.

The **M**/**m** interaction propensities, shown in Figure 2.12 pose a difficulty because **M** does not have an intrinsic maximal form. One can put as many **M**s as one likes in a Roman numeral and the only way to pre-determine how many **M**s there are is to limit the number of **M**/**m**s allowed in a Roman numeral. In this discussion the number of **M**/**m**s will be limited to four so that, with the carry, there are always exactly nine of **M**/**m**. The **M**/**m** interaction occurs strictly after the **D**/**d** interaction and is the necessarily last interaction of the Roman numeral addition. The interaction produces four of **M**/**m** and the differentness condition **Z** which indicates the completion of the addition.

Interaction propensities for **M,m**

```
[m,m,m,m,m,m,m,m,m] => [m,m,m,m,Z]
[M,m,m,m,m,m,m,m,m] => [M,m,m,m,Z]
[M,M,m,m,m,m,m,m,m] => [M,M,m,m,Z]
[M,M,M,m,m,m,m,m,m] => [M,M,M,m,Z]
[M,M,M,M,m,m,m,m,m] => [M,M,M,M,Z]
[M,M,M,M,M,m,m,m,m] => [M,M,M,M,Z]
[M,M,M,M,M,M,m,m,m] => [M,M,M,M,Z]
[M,M,M,M,M,M,M,m,m] => [M,M,M,M,Z]
[M,M,M,M,M,M,M,M,m] => [M,M,M,M,Z]
[M,M,M,M,M,M,M,M,M] => [M,M,M,M,Z]
```

Figure 2.12. Interaction propensities for M,m.

2.4.3. The new Roman numeral format

A modified Roman numeral is always exactly twenty conditions: four of **I** or **i,** one of **V** or **v**, four of **X** or **x**, one of **L** or **l**, four of **C** or **c**, one of **D** or **d**, and four of **M** or **m**. The number one is **mmmmdcccclxxxxviiiI**. The number zero is **mmmmdcccclxxxxviiii**. This is similar to a 32 bit 2s complement binary number which is always 32 bits regardless of its magnitude.

The Roman numerals.

MDCCCLXXXXVI and **CCCXXXXI**

now become

mmmMDCCCcLXXXXViiiI and **mmmmdCCCclXXXXviiiI**

and the addition becomes:

mmmMDCCCcLXXXXViiiI + **mmmmdCCCclXXXXviiiI** =
mmmMMdCCcclXXXxViiII.

The addition process accepts two completely represented numerals and produces one completely represented numeral preserving the numeral representation convention.

The above numerals are formatted for readability but there is still no intrinsic association differentiation.

mmmMDCCCcLXXXXViiiI = mimVXiMDXiCXmXCICcl = … = 1896

The only association relation is that all the differentness conditions are collectively associated within a single bounded place of common association.

2.4.4. Order from chaos

As shown in Figure 2.13 the indiscriminately associating differentness conditions of two Roman numerals, autonomously add themselves in a dependent progression of proximity of encounter completeness fulfillments and interactions to a necessarily last fulfillment and interaction which completes the sum and produces the condition **Z** singularly indicating the addition is completed. The **Z** condition might open the bag and spill the result or it might perform a coordination duty within the bag. The wavefront of interaction flows from top left to bottom right in Figure 2.13.

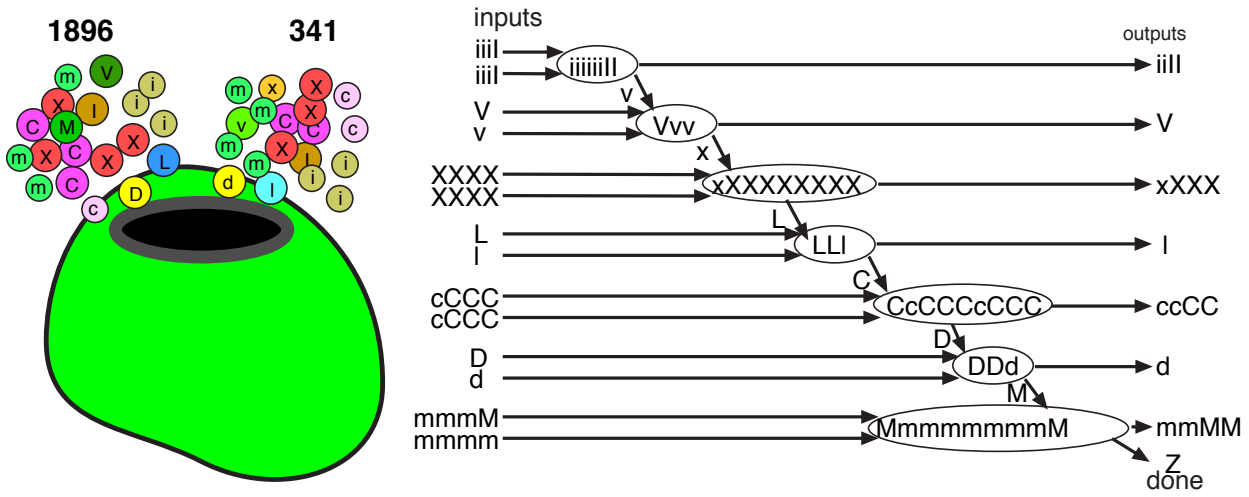


Figure 2.13. Modified Roman numeral addition in shaking bag.

2.4.5. The wavefront

Figure 2.14 illustrates the wavefront flow through a Roman numeral addition as result differentness conditions appear and interacting differentness condition disappear.

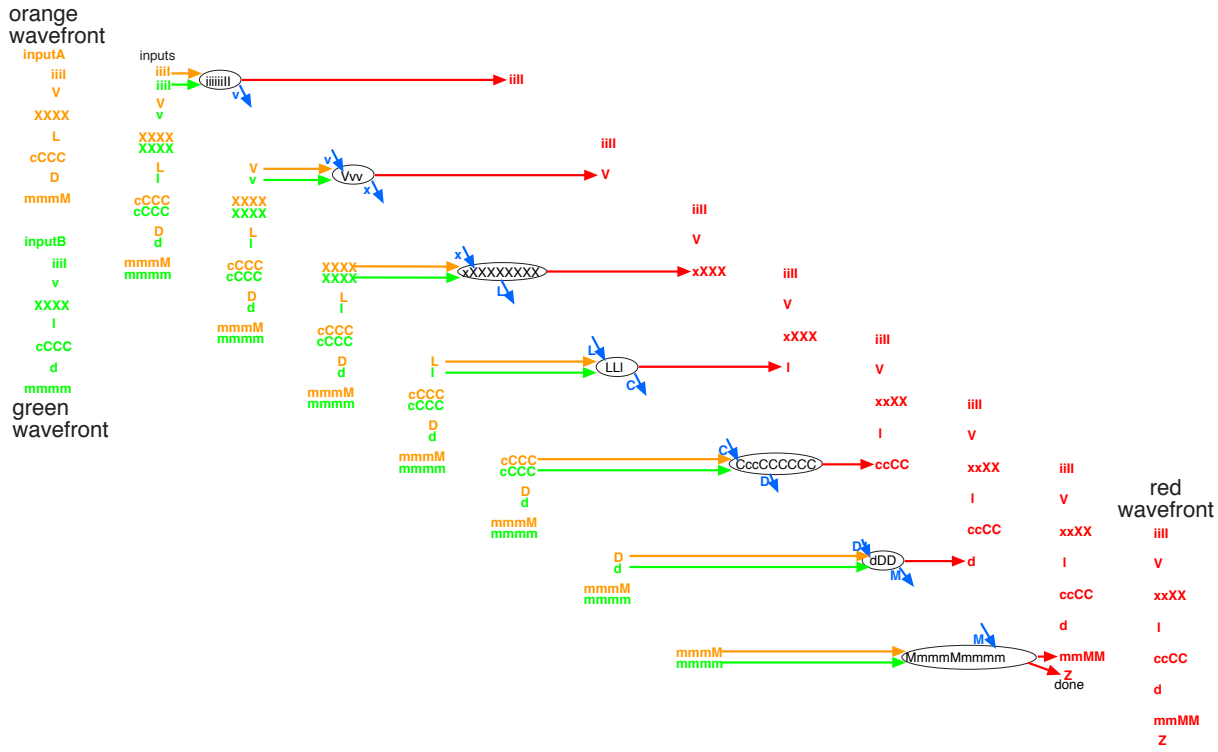


Figure 2.14. Wavefront flow through example Roman numeral addition.

The orange wavefront and the green wavefront interact and disappear. The blue wavefront is the intermediate carry wavefront which appears and disappears in the addition. The red result wavefront appears as the only remaining substantiality when the addition is completed. The red wavefront will flow on to a next interaction and disappear in its turn.

2.4.6. Collective interaction propensity

In Figure 2.7 Each **I** senses the presence of four other **I**s. Only when each **I** senses the presence of four other **I**s will they collectively interact producing a **V**. In Figure 2.13 each individual **I/i** senses the presence of 7 other **I/i**s. Only when each **I/i** senses seven other **I/i**s will they collectively interact. Similar to nuclear reaction critical mass. This collective behavior is contrived to make, as simply as possible, a particular example of spontaneous interaction in relation to a familiar arithmetical example.

In Appendix H. Roman numeral addition is further considered with interaction propensities solely in terms of condition differentnesses instead of in terms of quorum of presence. The examples are more complex and less contrived.

2.4.7. Isolation

The collection of differentness condition representing a number must remain isolated in a bag. If they escape the bag into a more general condition space they will disperse and no longer represent the number. They may flow from bag to bag but they cannot flow out of the isolation of a bag.

2.5. INTERLUDE: Pure condition differentiation

A pure condition expression is persistences asserting differentness conditions with different propensities of interaction indiscriminately encountering within a single place of common association (the shaking bag). The bag is a sameness, a oneness relating the many different persistences. Inside the bag there is no agency of central influence and there is no agency of explicit control. The persistences and their asserted conditions behave individually and independently. There is no ambiguous behavior that needs to be isolated or hidden. There is no need of any assistance or influence from outside the bag. It is a determined system of differentnesses (persistences asserting their differentness conditions) interacting according to interaction propensities within and abetted by the indeterminate chaos of the shaking bag instead of abetted by a mathematician with pencil and paper.

2.5.1. The search

The bag is not shaking to perform searches and realize interactions yet its shaking enables searches and realizes interactions. A persistence and its asserted differentness condition with its specific interaction propensity has no intrinsic agency to perform a search to fulfill its interaction propensity yet inside the shaking bag an asserted differentness condition will experience a journey of encountering many other differentness conditions with which it does not interact and then find a differentness condition with which it does interact.

2.5.2. Dependency, occurrence and the completeness criterion

Interaction begins with the occurrence of the interacting differentness conditions in the shaking bag. Dependency of interaction relations are expressed by the differentness of the conditions and their specific propensities to interact with other different conditions. An interaction occurs only with complete proximal encounter of interacting differentnesses which coordinates the progression of interaction behaviors. The occurrence of the result differentness conditions and the disappearance of the interacting differentness conditions indicates, that the different interacting conditions were completely proximally encountered, that the interaction has completed and that the result differentness conditions present the correct resolution of the interacting differentnesses.

There can be a multitude of pure condition interactions with mutually disjoint condition sets and interaction propensities concurrently realizing independent progressions of dependent condition interactions in a single frothing sea of differentness conditions within a shaking bag.

2.5.3. Place and time in the shaking bag

Interactions occur in their own time at their own place inside the bag. There is no referential where or when inside the bag. There is no consistent or coherent metric of temporal or spatial relations among the persistences, their conditions of differentness or among their interactions within the chaos of the shaking bag yet a fully determined unambiguous progression of interactions occurs. Trying to relate the behavior in the bag to an external metric of space or of time contributes nothing to either the understanding of or to the realization of the computation in the bag.

2.5.4. Natural pure condition differentiation

The familiar exemplar of pure condition differentiation is the warm cytoplasm of a biological cell filled with thousands of different proteins asserting differentness conditions with specific interaction propensities supporting the intermingled expression of thousands of different computations proceeding concurrently and continually without ambiguity. Each protein molecule is a persistence and the folding of each protein molecule manifests two or more configuration conditions of differentness determining its interaction with other molecular conditions of differentness.

2.5.5. A first principle fulfilled

Differentness conditions spontaneously and dependently interacting and changing (see second sentence of chapter) in the shaking bag is a complete and sufficient accounting in itself of its interactions in no need of any extrinsic support from outside the bag and with no ambiguous behavior in need of being isolated or hidden or remediated.

2.6. Wavefronts without design

The universe computes but it does not do arithmetic. This example is contrived and programmed to illustrate how computation with condition differentiation works. The shaking bag is the machine that realizes the program. The interaction propensities are the program. In uncontrived actuality interactions are indeterminate and incidental, just happening. It is clear enough how a shaking bag machine might arise but where can a dependency structure of interaction propensities come from?

2.6.1.1. Two interpenetrating domains of chaos

There are two interpenetrating domains of chaos that realize computational wavefronts. There is the chaos of multitudes of asserted differentness conditions with propensities of interaction (see section 2.2.1.1) and the chaos of their asserting persistences in the bounded space which ensures that all possible encounters of all asserted differentness conditions occur - cross association - and that all possible interactions happen. This enormous interaction possibility space is concurrently and rapidly traversed.

2.6.1.2. The wavefront

Among those interactions there will be some that are dependent on the previous occurrence of another interaction forming an incipient wavefront. An incipient wavefront might progress through many dependent interactions but eventually fail to find its next interaction and cease.

2.7. The wavefront cycle: The wavefront appreciating itself

But if the wavefront appreciates itself a greater substantiveness emerges. When a flowing wavefront happens to produce a differentness condition critical to an interaction two or more interactions prior in the wavefront the critical differentness condition closes a cycle of interaction forming a cyclic wavefront. The wavefront becomes dependent on itself and a limited population of differentness conditions. A wavefront cycle is sparked. Will it burn?

2.7.1. A cycle

A wavefront becoming a cycle is illustrated in Figure 2.15 based on the dependency network of Figure 2.3. There are interactions leading to the interaction that produces **a** then there are four interactions **A**, **B**, **C** and **D** of which **D** also produces **a** which enabling interaction **A** closes a cycle of interactions forming a cyclic network of interactions. Differentness conditions **a d g** and **m** are cycle differentness conditions connecting the interactions of the cycle network. Differentness conditions **b e h** and **k** are inputs to the network. Differentness conditions **c f i** and **n** are outputs of the cycle network.

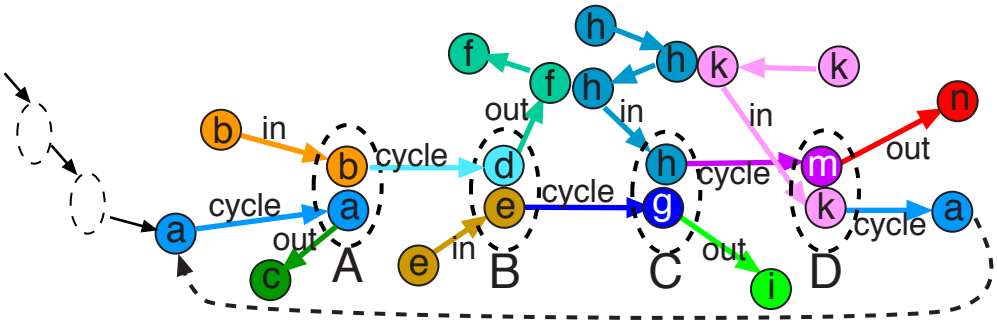


Figure 2.15. A wavefront becoming a cycle.

2.7.2. The cohering wavefront

The wavefront carrying the cumulative influence of the interactions of the cycle through condition space and differentnesses of time embodies and connects the cycle network while everything else of the cycle disappears. It is the wavefront that sparked the cycle and that continually manifests the cycle of interactions.

2.7.2.1. A fabric of wavefronts

The extrinsic differentness conditions flowing peripherally through a cycle could be components of other individual wavefronts even other cyclic wavefronts weaving a fabric of cycles.

2.7.3. Substantiveness

A quality of substantiveness was crossed when a wavefront became a cycle. The cycle is more robust than is a free wavefront because it is repeatedly flowing through interactions that were vetted by the wavefront before the cycle formed enabling the wavefront to flow robustly and indefinitely. A cycle possesses a memory that a free wavefront does not. The cycle overcomes the vicissitudes of a free wavefront focusing its wavefront in a continual sameness of condition space traversing the same network of dependency relations continually using the same differentness conditions over and over through differentnesses of time consuming the same input differentness conditions and producing the same output differentness conditions. The instantiation of a cycle wavefront forms a substantive ontological presence that was not

previously present, the origin of an object: a coherent locality of persistent liveness extending through condition space and through differentnesses of persistence time.

The cycle is a mechanism. It has an internal coherence that informs its continual succession of same interactions in relation to external objects which it consumes and produces.

> "Then one way to describe the project laid out above is that of developing an understanding of a material object as a "spatio-temporal chunk of reality that matters" --- thereby healing a temporary 300 year rift between matter and mattering." Brian Cantwell Smith†

The occurrence of a spatio-temporal orderliness in a background of spatio-temporal chaos is a spatio-temporal chunk of reality that matters.

2.7.3.1. Unregisterable

The formation of a cyclic dependency network is entirely incidental and once formed it continues to cycle on its own expressional merits as a most substantive emergence with nothing of greater substantiveness to appreciate it. Actually, nothing appreciates the cycle. The wavefront, which is flowing indefinitely by virtue of the cycle, cannot appreciate that it keeps flowing through the same interactions over and over. As far as the wavefront is concerned there is no cycle there is just an endless stream of next interactions. It is the continually disappearing and reappearing interactions of the cycle network that engenders the endless stream of next interactions but the interactions, continually disappearing, cannot appreciate that they are recurring over and over. Differentnesses of persistence time continually appearing and disappearing never recur and cannot appreciate anything about recurrence The differentness conditions with their interaction propensities appear and disappear with no sense of their roles. Even the immortal persistences working in the background as busy bees supporting all substantiveness cannot themselves appreciate the greater relations by which they are driven. There is nothing to speak for the cycle, not even its cohering wavefront. It just is: a circumstantial consequence of the differentness conditions with their propensities to interact and of the dependency relations among the interactions

2.7.3.2. Alone

It is the nature of substantiveness evolution that when a new substantiveness emerges it is alone as a most substantive substantiveness there is nothing to welcome it or judge its substantiveness, not even itself. If it happens to raise an overall level of substantiveness then it happens to raise an overall level of substantiveness. There is nothing intentionally trying to raise the overall level of substantiveness and there is nothing evaluating any overall level of substantiveness. A substantiveness such as the cycle emerges, does what it does and either persists in what it is doing or does not persist, survives or does not survive. Nothing more can be said.

There is always a most substantive substantiveness that is entirely on its own.

2.7.3.3. ineffable

There is no dialectic of meaning and mechanism because there is no meaning, no intentionality, no reference distal to the cycle itself. Survival is the only meaning and nothing

†. Smith, Brian Cantwell. Brian Cantwell Smith: From E&M to M&E: A journey through the landscape of computing. In Floridi, Luciano (ed.), *Philosophy of computing and information: 5 questions*, Automatic Press, 2008, p. 46

cares whether the cycle survives, not even the cycle. The cycle simply is and does what it does. nothing more can be said.

2.7.4. Cycle space

A wavefront cycle inherits exclusive singularity of space from the exclusive singularity of its succession of asserting persistences forming a coherent locality in condition space extending indefinitely through differentnesses of time.

2.7.5. Cycle time

The emergence of the cycle is the first circumstance in which time seems to have differentiating significance. Each cycle of the cycling wavefront is a sameness of interactions occurring in differentnesses of time but time is not differentiating the occurrence of the cycles. The cycling wavefront continually accumulates the persistence time of the cycle as it flows around the cycle but its accumulating time is insubstantial with no agency of influence or appreciation. Actually, there is nothing that spans the cycle as a whole that can appreciate the temporal repetition of the cycle (see section 2.7.3.1).

2.7.6. After meets before

The cycling wavefront represents the first instance of an after encountering a before. With each interaction the wavefront presents the interaction with a consequence of its previous instantiation.

2.7.7. Continuance of cycle interaction

Figure 2.16 shows on the left an established wavefront cycle network dependent on very specific differentness conditions very specifically interacting. On the right the network is redrawn circularly. For the cycle to continue all of the differentness conditions must be available for each interaction. What, for instance, if there is no **e** available when a **d** occurs? How can it be ensured that an **e** will be present when needed by **d** to interact?

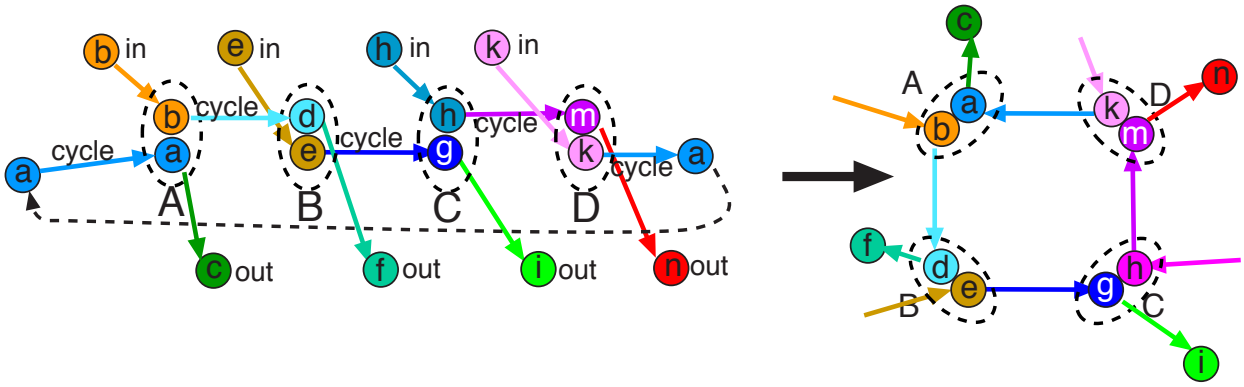


Figure 2.16. An established wavefront cycle and its circular representation.

2.7.8. Sufficiency of resource

Consider a diffusion gradient with a local concentration of input differentness conditions, **b, e, h** and **k**. The occurrence of the interactions **A**, **B**, **C** and **D** were established before the cycle closed. The interactions themselves produce the cycle dependency differentness conditions **a d g** and **m** enabling each interaction in turn. Once sparked the cycle will burn indefinitely, as long as there is a supply of input differentness conditions **b, e, h** and **k**, its oxygen. The cycle differentness conditions **a d g** and **m** will wait indefinitely for **b, e, h** and **k** conditions. The cycle

may slowly smolder but it will persist. When sufficient resources become available in the dense concentration of a diffusion gradient, for instance, it will burst into flame.

2.7.8.1. Consequences of sufficiency of resource

Consider that there will be multiple wavefronts pursuing the same progression of interactions through the same concentration of differentness conditions and engendering the same dependency cycle. Each cycle will be producing differentness conditions **a d g** and **m**. Assuming there is 100 wavefronts cycling and briefly assuming for a moooment synchrony among the wavefronts there will be 100 **d**s produced. These 100 **d**s become un-individuable free agents in the common supply and not individually registerable to the interactions that produced them. The cycles are indiscriminately sharing their **d**s. Each **d** will find a next interaction and its wavefront will continue but the next interaction cannot be registered to a specific previous interaction and hence cannot be registered to a specific wavefront. The wavefronts cannot be individually traced or otherwise individuated. All that can be said is that there are 100 wavefronts cycling. Maybe not even that. Perhaps all that can be said is that **b, e, h** and **k** conditions are becoming **c f i** and **n** conditions.

2.7.9. The wavefront blob

Disassuming synchrony among the wavefronts and referencing Figure 2.16 there will always be on average 25 each of **a**, **g**, **d** and **m** and all four interactions will commence occurring indiscriminately. The most structured way to speak of this is as four wavefronts pipelining through each of 25 cyclic dependency networks. There are still a 100 wavefronts cycling through a common structure of interaction dependency relations. But there is no registrable structure of dependency relationships in the chaos of wavefronts. The cycling wavefronts form a chaotic production line consuming differentness conditions **b e h** and **k** and producing differentness conditions **c f i** and **n**. Given the availability of interaction resources, the wavefronts and their dependency networks have intrinsically relaxed to maximally concurrent interaction behavior, a wavefront blob.

2.7.9.1. Nonspecific specificity

The most straightforward way to characterize the wavefront blob is that all interactions are happening all at once but this does not convey why they might be interacting all at once and all together, why they might continue indefinitely to interact or why they might actually be performing a deterministic process. It is the cyclic network of Figure 2.16 that pulls all of this together for us humans.

The notion of cycling wavefronts pipelining through a cyclic dependency network of differentness condition interactions got us to the wavefront blob but there is no registrable wavefront in the chaotic blob that corresponds directly to Figure 2.16. The ontological presence of the cycle that constructed the blob has disappeared within the ontological presence of the blob.

There is no way to trace an **n** back to any specific **k**. In fact there is no way to trace **n** back to **k** at all. There are four input differentness conditions and four output differentness conditions with no way, in the context of the blob, to relate any specific output differentness condition to any specific input differentness condition.

2.7.9.2. Constructible but not reducible

We traveled through the structure of the interaction to the structure of the wavefront to the structure of the cycle to get to the blob then all of the structure that constructed the blob falls apart within the chaotic blob. Following the primitives bottom up led directly to the blob but trying to go top down from the blob does not lead coherently to the primitives. The blob can be

easily constructed abstractly and actually but cannot be easily reduced either abstractly or actually.

Trying to extract the cycle view of a wavefront blob from independent distal observation can be challenging. However, such biological metabolism wavefront blobs have been discovered and characterized by distal humans which are described in textbooks of cell biology and presented similarly to Figure 2.16 as a cycle of dependent interactions with direct and peripheral dependency relations consuming inputs and producing outputs.

2.7.9.3. Ineffability

It should be evident by this point that cyclic wavefronts pipelining through 25 cyclic dependency networks is just a convenient way for us humans to try to think of the chaotic conflagration of interactions of cycling wavefronts. This is a story written by a human for humans trying to explain an autonomously constructing world with no humans in it.

We have to use pictures and words invented by humans and defined in a context of humans to tell the story. The actuality of the dynamic condition space can only be approximated. There are not actually cycle structures as in Figure 2.16, i.e., the persistences are not structured in a cycle. At this stage the persistences are all wandering freely in condition space. There are only the differentness conditions and their interaction dependency relations. Deterministic behavior in the midst of and abetted by nondeterministic chaos is not a common theme of human discourse.

It is not necessary that the denizens appreciate the big picture and where they fit into it. It is only necessary that each performs its part. It is immaterial to the denizens of the story that we might be thinking about them or that we even exist which at this stage we don't. From the emergence of the first blobs it will be a few billion years before the story includes the substantivenesses of humans assuming that everything is about them. There is a fundamental incommensurability of expressivity that this human story of the journey has to bridge. Hence the oft repeated phrase "nothing more can be said".

2.7.9.4. The blob space

Even though wavefronts cannot be individuated within a wavefront blob there is still substantiveness of wavefront flow within the blob. The wavefront blob inherits exclusive singularity of condition space from the exclusive singularity of its component cycling wavefronts which in turn inherit from their component persistences. The wavefronts form an exclusively singular, coherent but amorphous locality of condition space, a substantive ontological presence that is more substantive and less comprehendible than is an individual cycle.

2.7.9.5. The blob time

Referencing Figure 2.16 the blob can be viewed as a continually behaving interaction that consumes inputs **b e h** and **k** which are continually disappearing and produces outputs **c f i** and **n** which are continually appearing. The blob itself does not disappear as do primitive interactions but continually persists by virtue of its continually and independently cycling component wavefronts each imparting its own differentnesses of persistence time creating a chaos of unregisterable differentnesses of time in relation to the blob as a whole.

In one sense the flow of time has stopped in relation to the blob as a whole because the blob never transitions, never disappears. There is no singular before and after, no registrable begin or end, in relation to the blob as a whole, no instant of transition as there is with a single interaction. In another sense the differentnesses of time, flowing continually with indeterminate phase relations, are effectively and irreducibly flowing continuously. There are no intrinsically registrable boundaries of differentnesses of time in relation to the blob, i.e, no registrable ticks

and tocks. One might be tempted to suggest that blob time is flowing of itself equably without regard to anything external.

2.7.9.6. Substantiveness of the blob

The blob, with its chaos of cycling wavefronts, forms a self sustaining persistence of behavior at one coherent sameness of place in condition space extending indefinitely through differentnesses of time determinately consuming differentness conditions **b e h** and **k** and producing differentness conditions **c f i** and **n**. Wavefronts cycles and blobs remain active and coherent as long as there are sufficient resources present. The blob is a continually present ontological object.

The blob is a mechanism, a machine, with an internal coherence that informs its continual successive same behaviors in relation to external objects which it continually consumes and produces. It is an effective mechanism with no meaning, no intentionality, no semantic.

> In fact, it is metaphysically astonishing, to say nothing of deucedly lucky,
> That anything is effective --- that it is in virtue of the exemplification of
> any property that anything can have any causal consequence whatsoever.
> Brian Cantwell Smith‡

Being effective without intentionality, without causation I do not believe was credited.

2.7.9.7. Blob wavefront memory

If **b e h** and **k** differentness conditions diminish the cycle specific differentness conditions **a**, **g**, **d** and **m,** if they do not disperse, will persist waiting on the supply of **b e h** and **k** differentness conditions to return and commence the blob cycling. The lingering cycle differentness conditions **a**, **g**, **d** and **m** are halted wavefronts effectively remembering the blob and waiting to resume flowing.

2.7.9.8. Blob wavefronts

The differentness conditions disappearing into the blob may have come from one or more other blobs and the differentness conditions appearing out of the blob may go to one or more other blobs forming a blob wavefront or even a cycle of blobs. A blob wavefront is more substantial than a primitive interaction wavefront in that all parts of the wavefront connecting different blobs are continually present as a composite wavefront of many individual uncoordinated wavefronts. A cycle of blobs can in turn relax into an even more complex mess of seemingly random behaviors that are nevertheless performing a fully ordered deterministic process.

2.8. Statically conjoining persistences

Up to this point persistences have been only the indiscriminate carriers of discriminate differentness conditions facilitating all possible encounters of differentness conditions ensuring that all possible interactions occur. Two interpenetrating dimensions of expressivity. One dimension of discriminate specific relations and a complementary dimension of indiscriminate promiscuous relations, encounters, that facilitate the specificity of the discriminate specific relations.

Persistences also statically conjoin, statically encountered, motionless in relation to each other, compromising the dimension of indiscriminate promiscuous relations but establishing a

‡. Smith, Brian Cantwell, *Computational Reflections*. Cambridge, MA: MIT Press, 2026. p 342

dimension of statically associated discriminate relations. Asserted differentness conditions can now interact only with directly associated persistences with which they may or may not interact. A conjoinment can be either active with its asserted differentness conditions interacting or inactive with its asserted differentness conditions not interacting.

How or why persistences might conjoin is a different story about physics and chemistry for a different teller. This journey is trying to tell the story of computation without distraction. The reason that persistences conjoin may not be computational but their possibility of conjoining is computational.

2.8.1. Conjoinment subspace

A conjoinment of persistences as a whole inherits unique singularity of condition space from the totality of the unique spatial singularities of its component persistences.

A conjoinment of persistences also inherits from the individual spatial singularities of its component persistences an internal structure of "Place in relation to": a structured subspace of persistent and determinate dependency relations residing within the transient indeterminacy of the greater condition space which has no places and no place relations: a coherent ontological object more substantive than an individual persistence.

2.8.1.1. Conjoinment subspace structure

Conjoinment of persistences can be construed in a first approximation as stacking balls. In Figure 2.17 from left to right three balls **A**, **B**, and **C** conjoin with each conjoined to the other two. A fourth ball **D** can conjoin with all four being mutually conjoined, mutually interactively encountered, forming an interaction cluster in which interaction will be determined by the asserted differentness conditions just as it is with free persistences but now the persistences are pre-encountered and will not wander off. A fifth ball **E** can conjoin with three of the balls but cannot conjoin with all four and will be isolated from - not interactively encountered with - one of the other balls in this case **C**. There are now two interpenetrating interaction clusters **A**, **B**, **C**, **D** and **A**, **B**, **D**, **E** sharing persistences.

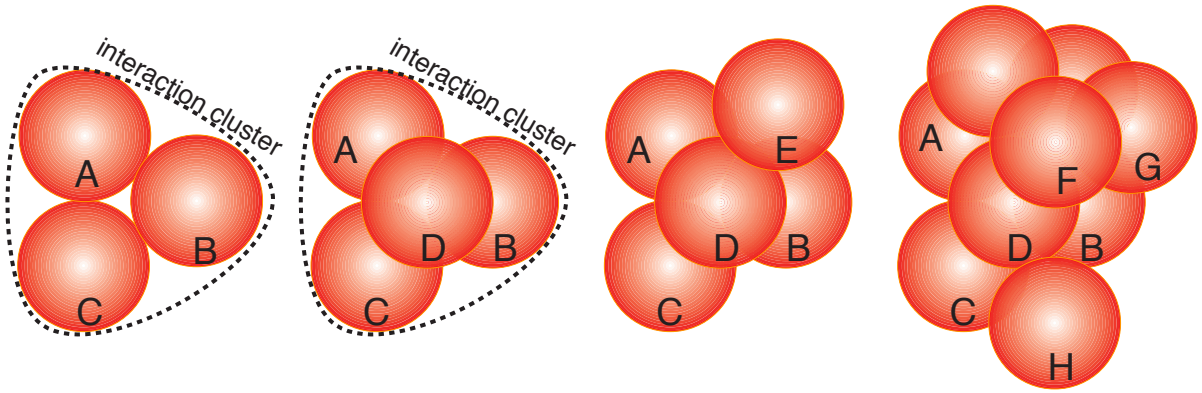


Figure 2.17. Persistence view of conjoinment structures.

As more balls **F**, **G** and **H** conjoin there will be more and more balls isolated - not in direct conjoinment - not interactively encountered. Balls **F** and **G** are isolated from **C** and **H**.

There will also be more mutually conjoined interaction clusters and the clusters will overlap. Each ball will be surrounded by and will be a member of clusters of interactive proximity and isolated from all balls beyond those neighbor clusters.

Within an interaction cluster interaction is determined by asserted differentness conditions. All interactions occur within an interaction cluster. Interaction clusters overlap and

interpenetrate. Wavefronts of interaction flow through a conjoinment from cluster to cluster forming paths of wavefront flow throughout a conjoinment.

2.8.2. A new expression of differentness: Isolation

A new expression of differentness emerges with conjoinment.

2.8.2.1. Isolation of conjoinment

For a differentness condition free in condition space its interaction propensity is its one meaning throughout condition space in which persistences asserting the same differentness condition interchangeably perform the same interaction. Differentness conditions within a conjoinment are constrained to interact only with the differentness conditions of directly associated conjoined persistences. Differentness conditions asserted within the conjoinment are isolated from and different by virtue of their isolation from all same differentness conditions asserted outside the conjoinment.

2.8.2.2. Isolation within conjoinment

Same differentness conditions asserted in different isolated places within a conjoinment association structure are differentiated by the isolation differentness of their places within the conjoinment structure. Interactable differentness conditions asserted at isolated places within a conjoinment will not interact. Same differentness condition asserted at isolated places within a conjoinment are unambiguously different by virtue of their isolation.

2.8.2.3. Association differentiation

This differentness of isolation within a conjoinment is association differentiation.

The story of conjoining persistences is about the emergence of association differentiation. A story of fully emerged association differentiation, (computation expressed entirely within a single conjoinment) is presented in Chapter 3 and Chapter 4.

2.8.2.4. Extended differentness

Isolation of conjoinment and isolation within conjoinment indefinitely extends the expressivity of the primitive differentness conditions beyond what they can express in free condition space. The extension is limited only by the availability of persistences asserting differentness conditions.

2.8.2.5. A new dimension of expressivity

Conjoinment introduces a dimension of discrete determinism in the midst of continuous chaos: a new dimension of specifically associated deterministically interacting persistences: a dimension in which nothing moves within a dimension in which everything moves: a dimension in which the only behavior is the transitioning of differentness conditions.

2.8.2.6. The ineffability of conjoinment

The conjoinment of the persistences cannot be characterized because persistences are not individuated apart from their asserted differentness conditions. It cannot be distally said that persistence A conjoined with persistence B and there can be no propensity to specifically conjoin among the persistences. Persistences conjoin as they conjoin and what happens happens.

2.8.3. Two persistence conjoinment

Figure 2.18 shows two persistences each with two differentness conditions that it can assert.

/{h m}

/{g k}

Figure 2.18. Persistences with two differentness conditions.

Figure 2.20 show the two conjoined persistences asserting differentness condition with propensity to interact. The arrows connecting differentness conditions indicate a persistence transitioning between assertion of its differentness conditions.

Figure 2.19. Two interaction - two persistence conjoinment.

Interaction propensities	persistence differentness conditions
[h g] => [m k]	/{h m}
[m k] => [h g]	/{g k}

Figure 2.20. The symbolic view.

2.8.3.1. The expressions

The expression **[h g] => [m k]** means that **"all of"** differentness conditions **h** and **g** interact producing **"all of"** differentness conditions **m** and **k**. The expression **/{h m}** means that there is a persistence that can assert **"one of"** differentness conditions **h** and **m**. The differentness conditions assertable by each persistence and the interaction propensities of the differentness conditions together sufficiently characterize the behavior of the conjoinment.

2.8.3.2. Conjoinment behavior

The interaction behavior of the conjoinment will depend on the initial configuration of asserted differentness conditions at conjoinment.

initial configuration	wavefront flow	initial configuration	wavefront flow
hk	deadlock	mk	alternating
gm	deadlock	gh	alternating

In Figure 2.20 if the initial differentness conditions are **m** and **k** then **m** and **k** interact producing **h** and **g** which with their persistences conjoined will immediately interact producing **m** and **k** which with their persistences conjoined will immediately interact producing **h** and **g** which with their persistences conjoined will immediately interact forming a continually flowing cyclic wavefront within the conjoinment.

The table indicates that of the persistences that conjoin asserting interactable differentness conditions 50% will be active and 50% will be inactive.

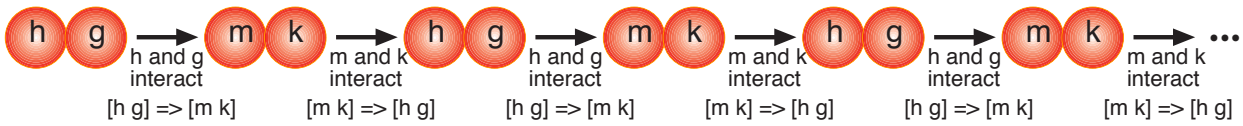


Figure 2.21. The interaction progression of the two persistence conjoinment.

The two persistence conjoinment above forms a substantiveness with isolated interaction behavior, a coherent piece of space that does not intrinsically extend beyond itself but that is free to wander within condition space encountering free differentness conditions.

2.8.4. Three persistence conjoinment

Figure 2.22 is a conjoinment of three persistences each asserting two differentness conditions with propensity to interact.

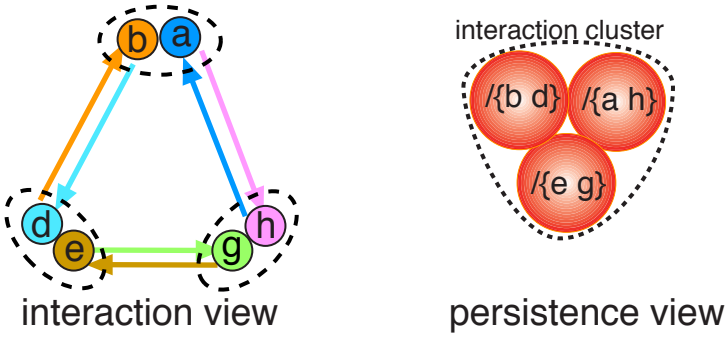


Figure 2.22. Three interaction - three persistence conjoinment.

Interaction propensities	persistence differentness conditions
[b a] => [d h]	/{h a}
[d e] => [b g]	/{b d}
[g h] => [e a]	/{e g}

Figure 2.23. The symbolic view.

2.8.4.1. The conjoinment behavior

The interaction behavior of the conjoinment will depend on The initial configuration of asserted differentness conditions at conjoinment determines the progression of interactions which can be seen as clockwise or counter clockwise in relation to the interaction view of Figure 2.22.

initial configuration	wavefront flow
beh	deadlock
adg	deadlock
bae	counter clockwise

initial configuration	wavefront flow
deh	counter clockwise
dea	clockwise
ghb	counter clockwise

bag clockwise ghd clockwise

If the initial configuration is **b**, **a** and **e** then **[b a]** will be the first interaction producing **[d h]**. Then **[d e]**will be the second interaction producing **[b g]**. Then **[g h]** will be the third interaction producing **[e a]** reestablishing the initial configuration **b a** and **e**. Then interaction **[b a]** starts the cycle over again with the interactions rotating counter clockwise around the network of Figure 2.22. If the initial configuration was **b**, **a** and **g** the rotation would be clockwise in relation Figure 2.22. There are always three differentness conditions asserted in the conjoinment.

The table indicates the possible behaviors of the conjoinment with each possible initial configuration. 25% of conjoinments will deadlock with no activity. 75% of conjoinments will result in an active network that is self sustaining and will cycle indefinitely.

2.8.4.2. Symbolic view of interaction flow

The interaction cycle in terms of the differentness conditions.

b a e	[b a] => [d h]	a and b interact	
d **h** e	[d e] => [b g]	d and e interact	
b h g	[g h] => [e a]	h and g interact	
b a **e**	[b a] => [d h]	a and b interact	back to initial configuration
d **h** e	[d e] => [b g]	d and e interact	

2.8.4.3. Persistence view of interaction flow

Figure 2.24 illustrates the flow of interaction for the three persistence conjoinment beginning with initial configuration b a e.

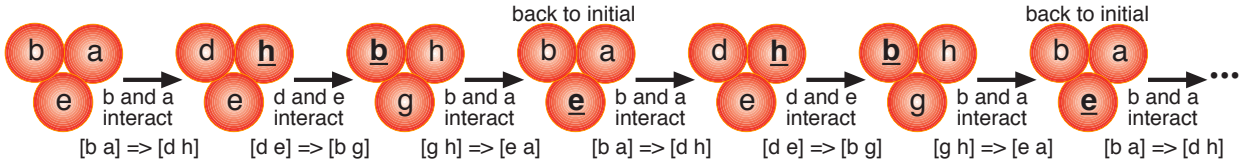


Figure 2.24. The interaction progression of the three persistence conjoinment.

2.8.4.4. Not very effable

The left of Figure 2.22 presents the conjoinment as a cycle structure and its behavior as an interaction wavefront rotating around the cycle. The right of Figure 2.22 presents three persistences all mutually encountered with no intrinsic orientation or motion. The symbolic expressions of Figure 2.23 characterize the differentness conditions and their dependency relations but not structure or dynamic behavior.

The notion of clockwise and counterclockwise depends on viewing the conjoinment from the front or the back but there is no orientation in condition space and the conjoinment does not present a front or a back or right or left. A conjoinment is only active or not active. There no orientation of activity. There is only the progression of interactions transitioning differentness conditions as in section 2.8.4.2 above and Figure 2.24. Wavefronts of transitioning differentness conditions flow through the spatial stillness of the conjoinment.

The universe does not try to understand what it is doing it just does. It is us humans trying to understand that need these several textual and graphic views of a conjoinment cycle from different perspectives to speak about what is going on. We are trying to project our own dialect of understanding onto a universe that is not constructed within any context of comprehensibility.

2.8.4.5. The conjoinment wavefront/bubble counterflow

As a wavefront flows through an interaction the interacting differentness conditions disappear result differentness conditions appear. For a wavefront to cycle through same interactions in a conjoinment the differentness conditions that disappeared with the interaction must reappear for the next occurrence of the interaction. The conjoined persistences must provide the reappearing differentness conditions to sustain the wavefront flow within the conjoinment.

For each interaction two interacting differentness conditions disappear and two result differentness conditions appear. One result differentness condition will be the wavefront result differentness condition that proceeds to the dependently next interaction. The other result differentness condition will be a bubble result differentness condition that reenables the next occurrence of the dependently previous interaction and waits for the wavefront to flow to it. The wavefront differentness conditions and bubble differentness conditions can be said to flow counter to each other.

Bubbles are not required in free condition space where there might be many encounterable instances of any particular differentness condition. With the free condition space cycle of Figure 2.15 there is only wavefront result differentness conditions within the closed form of the cycle. There are no bubble result differentness conditions. The interaction enabling differentness conditions come from condition space, outside the dependency structure of the wavefront.

In the interaction flow presentations the bubble result differentness conditions are **bolded** and underlined as in section 2.8.4.2.

2.8.5. Four persistence conjoinment

Figure 2.26 is a conjoinment of four persistences each asserting two differentness conditions with propensity to interact.

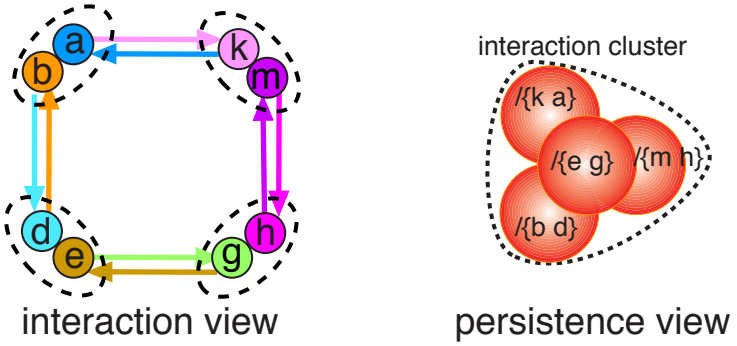


Figure 2.25. Four interaction - four persistence conjoined cycle.

Interaction propensities	persistence differentness conditions
[b a] => [d k]	/{k a}
[d e] => [b g]]	/{b d}
[g h] => [e m]	/{e g}
[k m] => [a h]	/{m h}

Figure 2.26. The symbolic view

2.8.5.1. The conjoinment behavior

In reation to the interaction view the interactions will deadlock, rotate or blink depending on the initial configuration of asserted differentness conditions of the four persistences at conjoinment.

initial configuration	wavefront flow	initial configuration	wavefront flow
amgd	deadlock	ghkb	counter clockwise
behk	deadlock	ghkd	blink
abeh	counter clockwise	ghab	blink
abem	blink	ghad	clockwise
abgh	blink	mkbe	counter clockwise
abgm	clockwise	mkbg	blink
dehk	counter clockwise	mkde	blink
deha	blink	mkdg	clockwise
demk	blink		
dema	clockwise		

The table indicates the possible behaviors of the persistence conjoinment with each possible initial configuration. 12.5% of conjoinments will deadlock with no activity. 87.5% of conjoinments will result in an active network that is self sustaining and will cycle indefinitely. There are 16 different initial configurations. Because of the way the table was generated there are two duplicates in the table which are color coded.

2.8.5.2. The probabilities of active conjoinment

The probabilities of active conjoinment have been explored to indicate that there are definite probabilities of active conjoinment. There is the probability of the conjoining persistences asserting appropriate differentness conditions. After that the tables indicate the probability of the conjoinment being active.

2.8.5.3. Interaction flow behavior

If the initial configuration is **b**, **a**, **e** and **h** then **[b a]** will be the first interaction producing **[d k]**. Then **[d e]**will be the second interaction producing **[b g]**. Then **[g h]** will be the third interaction producing **[e m].** Then **[k m]** will be the fourth interaction producing **[a h].** reestablishing the initial configuration **b a**, **e** and **h**. Then interaction **[b a]** starts the cycle over again with the interactions rotating counter clockwise in relation to the interaction view of Figure 2.25. If the initial configuration was **b**, **a**, **g** and **m** the rotation would be clockwise. There are always four differentness conditions asserted in the conjoinment.

A blink behavior is interactions **a b** and **g h** occurring together then interactions **d e** and **k m** occurring together then interactions **a b** and **g h** occurring together then interactions **d e** and **k m** occurring together and so on.

2.8.5.4. Symbolic view of interaction flow

There is no isolating structure of conjoinment (see the second from left figure of Figure 2.17) and there is no behavioral orientation in condition space. There is only an order of progressing interactions. Again the bubble result differentness conditions are **bold**.

d e h k	[d e] => [b g]	d and e interact	
b g h k	[g h] => [e m]	g and h interact	
b **e** m k	[k m] => [a h]`	m and k interact	
b e **h** a	[b a] => [d k]	a and b interact	
d e h **k**	[d e] => [b g]	d and e interact	back to initial configuration

The wavefront flow is **g, m, a, d, g, m ,a, d** ….

For blink, the result conditions are interchangeably wavefront or bubble. All result differentness conditions interact immediately.

a b g h	[b a] => [d k]	[g h] => [e m]	a and b interact and g and h interact
k d e m	[d e] => [b g]	[k m] => [a h]	d and e interact and k and m interact
a b g h	[b a] => [d k]	[g h] => [e m]	a and b interact and g and h interact

back to initial configuration

2.8.5.5. The persistence view of interaction flow

This is another way to visualize how the four persistence conjoin of section 2.8.5 looks and behaves, Figure 2.27 shows the differentness condition interaction transition in the context of the persistence view. The letter on each persistence is the differentness condition that the persistence is asserting. The persistences themselves are not individuable (no structural isolation) apart from their asserted differentness condition. The sequence of interaction is the same as in 2.8.5.4.

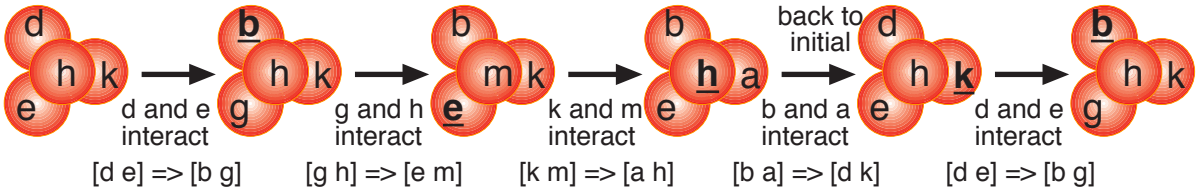


Figure 2.27. The interaction progression of the four persistence conjoinment.

The bubbles are bolded and underlined. The wavefront flow is **g, m, a, d, g, m ,a, d** ….

2.9. Conjoinment in condition space

A conjoinment is free in condition space to encounter free differentness conditions with which it may interact.

2.9.1. Two persistence conjoinment in condition space

Consider that **m** and **k** interact with each other but that **h** and **g** do not interact with each other and interact only with differentness condition **o** which is not asserted by one of the conjoined persistences, Figure 2.28. When a differentness condition **o** is encountered the interaction occurs. Differentness condition **o** transitions to **p** with its asserting free persistence unencountering. Differentness conditions **h** and **g** transition to **m** and **k** which immediately interact transitioning to **h** and **g** awaiting a next **o**. With each interaction the conjoinment immediately resets and begins looking for a next **o**.

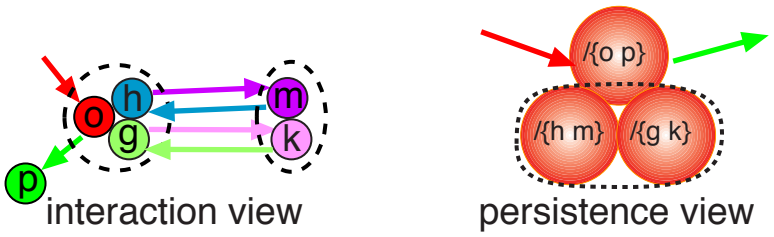


Figure 2.28. Composite persistence with composite differentness condition

Interaction propensities	persistence differentness conditions
[h g o] => [m k p]	/{h m}
[m k] => [h g]	/{g k}
	/{… o p ...}

Figure 2.29. Symbolic view.

2.9.2. Catalytic iteration

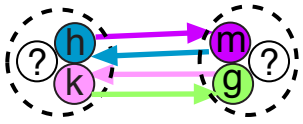
The two persistence conjoinment of Figure 2.28 wanders through condition space asserting its **hg** composite differentness. upon encountering a differentness condition **o** it interacts and immediately reasserts the **hg** differentness condition looking for a next **o** differentness condition with which to interact successively turning differentness condition **o** into differentness condition **p**. The conjoinment acts as a catalytic agent continually and repeatedly turning **o**s into **p**s.

2.9.3. Composite persistence asserting composite differentness conditions

With its inherited wholeness of spatial singularity the two conjoined persistences forms a composite persistence within its condition space. The differentness conditions **h** and **k** always appear together and **m** and **g** always appear together and constituting composite differentness conditions with new interaction propensities asserted by the composite persistence.

The conjoinment is deadlocked asserting **h** and **k** at initial conjoinment but it is free to wander through condition space with the opportunity of encountering a differentness condition with propensity to interact with composite differentness condition **hk** as a threesome. If that interaction occurs the conjoined persistences transition to **g** and **m** and the conjoinment is deadlocked again. If a differentness condition is found to interact with composite differentness condition **gm** then **g** and **m** will transition to **h** and **k**. Like a primitive persistence with two differentness conditions the conjoinment will indefinitely perform the same two interactions alternately in condition space.

The conjoinment is acting as a composite persistence asserting two composite differentness conditions, **/{[h k] [g m]}** as in Figure 2.30. The conjoinment may or may not find interaction in condition space.



Interaction propensities	persistence differentness conditions
[h k ?] => [g m ?]	/{[h k] [g m]}
[g m ?] => [h k ?]	/{…?… }
	/{… o p ...}

Figure 2.30. Composite persistence with composite differentness conditions.

2.9.4. Four persistence conjoinment in condition space

A conjoinment is free to wander in condition space and can extend beyond itself by interacting with extrinsic differentness conditions not asserted by its conjoined persistences. Any interaction of the four conjoinment cycle network might require a third differentness condition from the free flowing differentness conditions of condition space.

Figure 2.31 illustrates the four persistence/four interaction cycle of Figure 2.26 with four different configurations of extrinsic interaction. The four persistences and their asserted differentness condition will keep the cycle intact and active while the conjoinment as a whole wanders through condition space searching for the extrinsic differentness conditions.

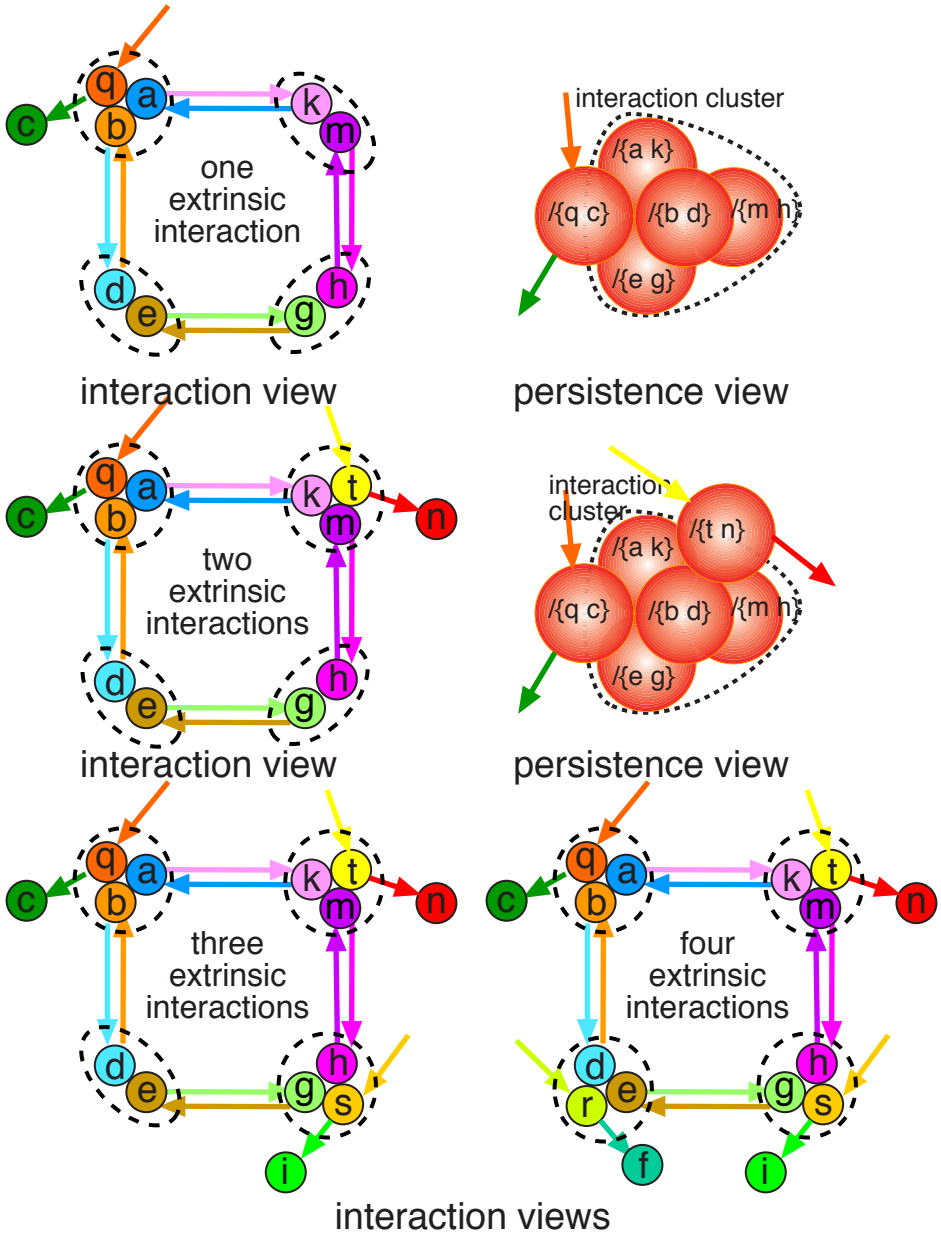


Figure 2.31. Four interaction conjoinment cycles with extrinsic interactions.

2.9.4.1. Catalytic iterative ordering

The cyclic conjoinment forms a composite persistence that asserts one composite differentness condition at a time with a propensity to interact with extrinsic differentness conditions. It is a catalyst repeatedly imposing sequential ordering on its extrinsic interactions with its cycling wavefront.

2.9.4.2. Wavefront memory

At each extrinsic interaction the internal cycling wavefront stops and its result differentness condition waits indefinitely until the extrinsic differentness condition is encountered and the interaction occurs. It might be construed as the wavefront remembering its wavefrontness responsibilities through differentness of time. But more properly there has been no differentness of time. The differentness conditions waiting on the interaction to occur exist in a same era of time of each asserting persistence. There has been no differentness of time, no advance of time, no passage of time for the persistences and their asserted differentness conditions, i.e., no passage of time for the waiting wavefront to remember through. Time will advance, transition, become different for the waiting persistences when the interaction occurs and the asserted differentness conditions transition.

This property of a stopped wavefront waiting indefinitely in its single differentness of persistence time can be used to form a memory in relation to other passages of time. While the wavefront waits its conjoinment retains its structure and behavior indefinitely remembering itself. This is in contrast to the blob which retained it behavior but not its structure. While the blob was expressed purely in terms of differentness conditions and their interaction propensities with no intrinsic association structure, a conjoinment expression includes the constraints of its association (conjoinment) structure which which is retained forming the memory context of the waiting wavefront.

The conjoinment still wanders as a whole in condition space searching for interaction but now it wanders around with a memory of what the next interaction is, i.e., the interaction is dependent on the remembering wavefront as well as particular differentness conditions. Is there an emergent intentionality? Having emerged incidentally the conjoinment with its flowing wavefront ensures a specific sequence of specific interactions.

The waiting wavefront is used to realize the addressable memory of section 4.10. A population of related wavefronts indexically stopped awaiting a next interaction in an indexical conjoinment structure becomes an addressable data structure: information. The reflowing interaction, read, is facilitated with the encounter of an index differentness condition arriving in a context of persistence times flowing distally to the waiting wavefront.

An indexical structure of stopped wavefronts, information, might, for instance, be a book. The words and sentences are the stopped wavefronts. The title is first index. The pages numbers are a next index. The structure of the words on the page are a final index. The wavefronts are indexed and reflowed by a substantiveness called a reader.

2.10. Interlude: The computation of evolution

This dance of opportunity between conjoinment of persistences (static association relations) and the interaction propensities of differentness conditions (dynamic interaction relations) in the context of condition space facilitating greater and greater substantiveness extending further and further into condition space and through differentnesses of persistence times is evolution.

The universe computes spontaneously, probabilistically and opportunistically. Try everything. What works works. Humans compute deliberately, determinately and intentionally. But, how is it that a human can be deliberate, determinate and intentional? How is it that indeterminate circumstance can turn into deliberate intention?

2.10.1. Evolution's ratchet: a universe learning

There is no intentionality guiding the occurrence of greater substantiveness. There is only a probability of substantiveness incidentally occurring and persisting in a space of chaotically interacting differentness conditions. Nothing more. The probabilities do not have to be great. Once a robust self contained and self sustaining substantiveness occurs it becomes a ratchet step making the possible occurrence of greater substantiveness more likely. Each step of ratchet survival is a lesson learned and remembered by the universe. Tiny but sufficient probabilities of anti entropy if you like. Evolution is about survival all the way down.

> "and thereby, perhaps at least partially to understand---how a structured lump of clay can sit up and think." Brian Cantwell Smith[§]

2.11. Conjoinment of conjoinment

Conjoinment can conjoin by dependently coupling their flowing wavefront flow through a shared internal persistence or by dependently linking wavefront flow of conjoinments through a shared external persistence or by continuing wavefront flow from conjoinment to conjoinment.

2.11.1. Dependently coupling wavefront flow

Complex conjoinments can be formed using a small set of differentness condition over and over in isolation.

In Figure 2.32 two cycle conjoinments are further conjoined by coupling their wavefronts through persistence **/{x w}**. The two cycle conjoinments are differentiated by their use of unique differentness conditions.

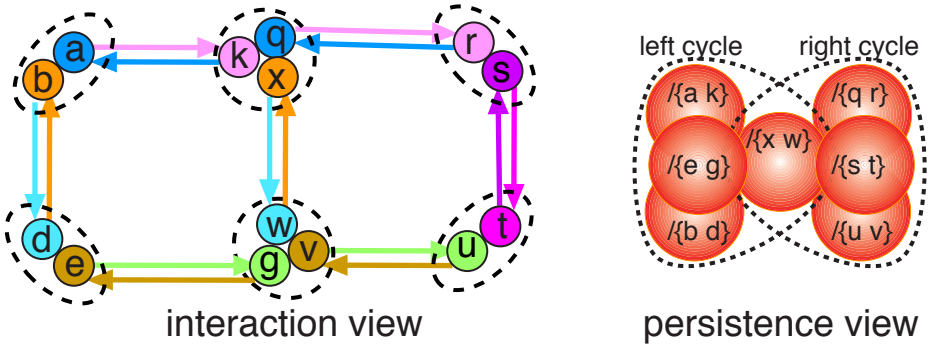


Figure 2.32. conjoinment cycles sharing their internal wavefronts.

The coupled wavefronts must flow in the same direction through the shared persistence. The wavefronts of the coupled conjoinment cycles must flow in opposite rotations such as with one flowing clockwise and the other flowing counterclockwise so that the two wavefronts flow in the same direction through the shared persistence **/{x w}**.

§. Smith, Brian Cantwell, *On the Origin of Objects*. Cambridge, MA: MIT Press, 1996. p 75-76

2.11.2. Dependently linking wavefront flow

In Figure 2.33 three identical cyclic conjoinments **A**, **B**, and **C** are linked through **/{o p}** persistences. The single persistences isolate the cycle interaction clusters and the interaction clusters isolate the single persistences. The wavefront of each cyclic conjoinment is dependent on the interaction with **o** and **p** but the wavefronts are not shared, are flowing independently and can individually flow in any of the 14 active configurations. There is a wavefront individually flowing around each cycle and there is an individual wavefront flowing back and forth in the **/{ p o}** persistence. All of these independently registrable wavefronts are coordinated by their interaction dependencies.

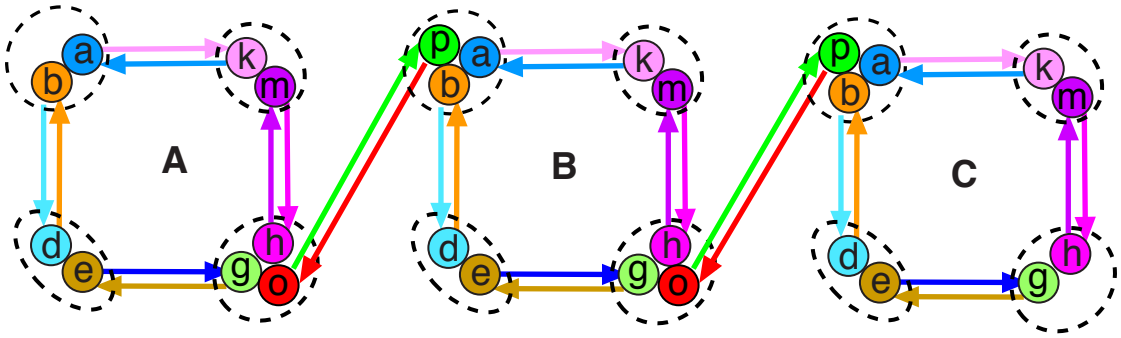


Figure 2.33. Conjoinments linked by conjoined persistence: interaction view.

Figure 2.33 illustrates isolation within conjoinment. Each four cycle conjoinments is expressed with its own persistences but they all share the same asserted differentness conditions and are linked with persistences asserting the same differentness conditions **/{o p}**.

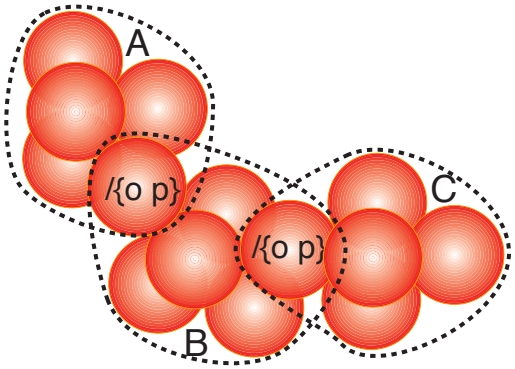


Figure 2.34. Linked conjoinments: persistence view.

2.11.2.1. Extending the expressivity of differentness conditions

The places of isolation texture the structure of a conjoinment enabling among other things indefinitely extensive construction of conjoinment with a limited set of differentness conditions reusing the differentness conditions over and over in the different isolated localities of the conjoinment extending the expressivity of the differentness conditions. There is an unlimited supply of isolated localities of conjoinment differentness.

2.11.3. Tiling conjoinment

A conjoinment unit can be larger. The conjoinment of Figure 2.35 is a two dimensional tile of conjoinment. Interactions **A**, **B**, **C** and **D** are labeled in the interaction view. Each interaction corresponds to an interaction cluster in the persistence view which is correspondingly labeled.

2.11.3.1. The cyclic wavefront

The wavefront flows around the cycle from interaction to interaction through persistences **/{5 6}**, **/{15 16}**, **/{11 12}** and **/{7 8}**.

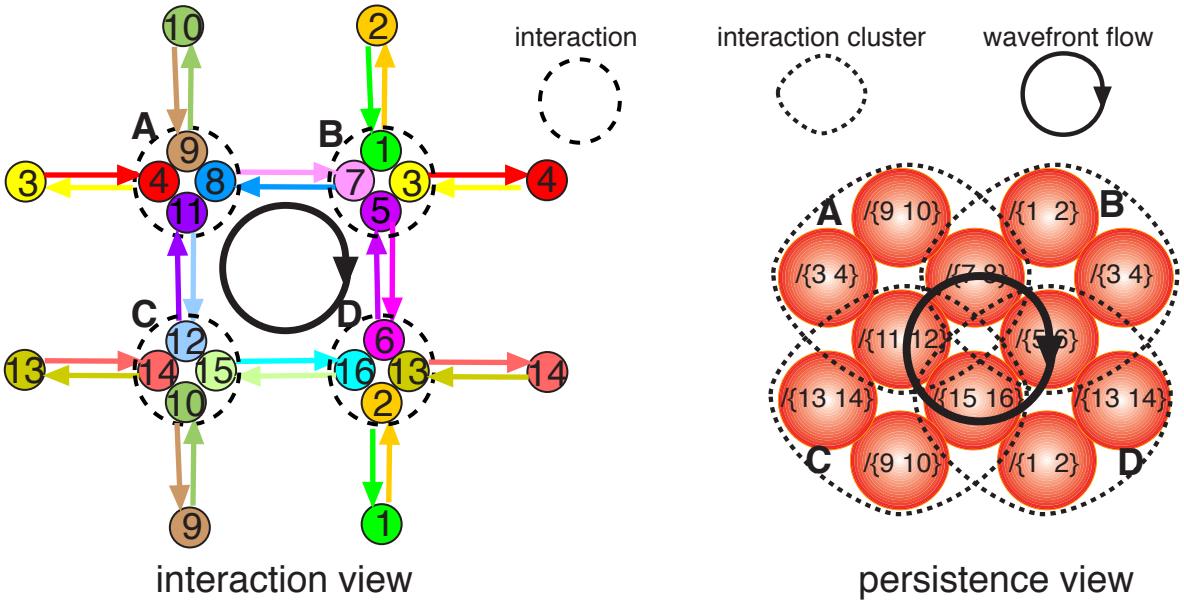


Figure 2.35. Tileable conjoinment.

Interaction propensities	persistence differentness conditions
[4 8 9 11] => [3 7 10 12]	/{1 2}
[1 3 5 7] => [2 4 6 8]	/{10 9}
[2 6 13 16] => [1 5 14 15]	/{3 4}
[15 10 12 14] => [16 9 11 13]	/{13 14}
	/{7 8}
	/{5 6}
	/{15 16}
	/{11 12}

2.11.3.2. The isolation

Opposing edges of the conjoinment contain identical persistences but the presistences are isolated by interaction propensities. For instance of persistence **/{3 4}** condition **3** interacts with conditions **7 1 5** but not with conditions **9 8 11**. Condtion **4** interacts with conditions **9 8 11** but not with conditions **1 7 9**. Similarly for persistence **/{7 8}** condition **7** interacts with conditions **3 5 1** but not with conditions **4 9 11** while condition **8** interacts with conditions **4 9 11** but not with conditions **3 5 1**.

The wavefront is sequencing the interactions adding to the unambiguous interaction protocol.

2.11.3.3. The surface tile

If the right persistences and the bottom persistences are removed as in Figure 2.36 a tileable conjoinment emerges.

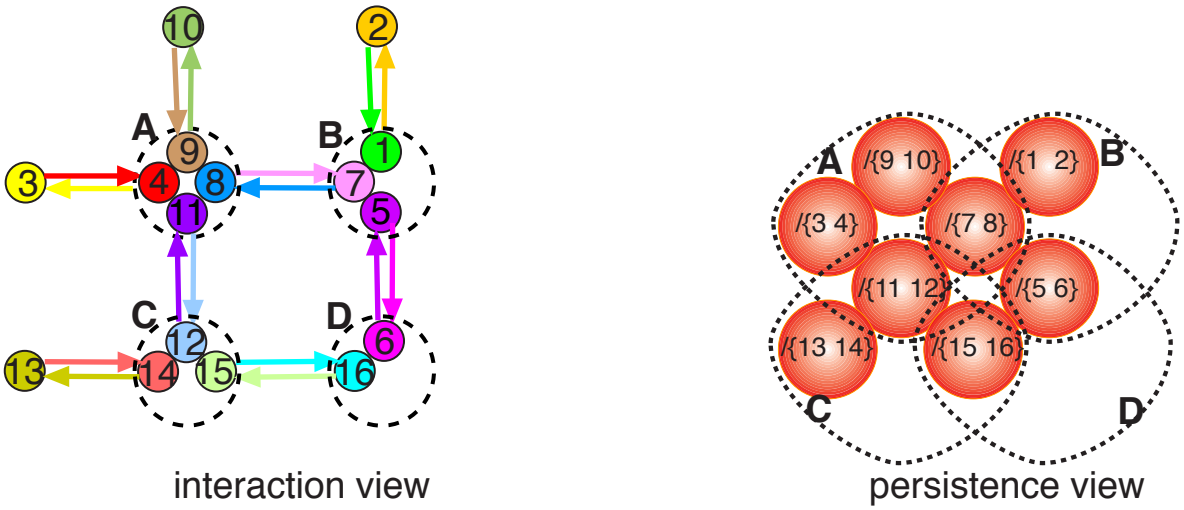


Figure 2.36. Surface tile

Figure 2.37 shows four tiles positioned to be tiled.

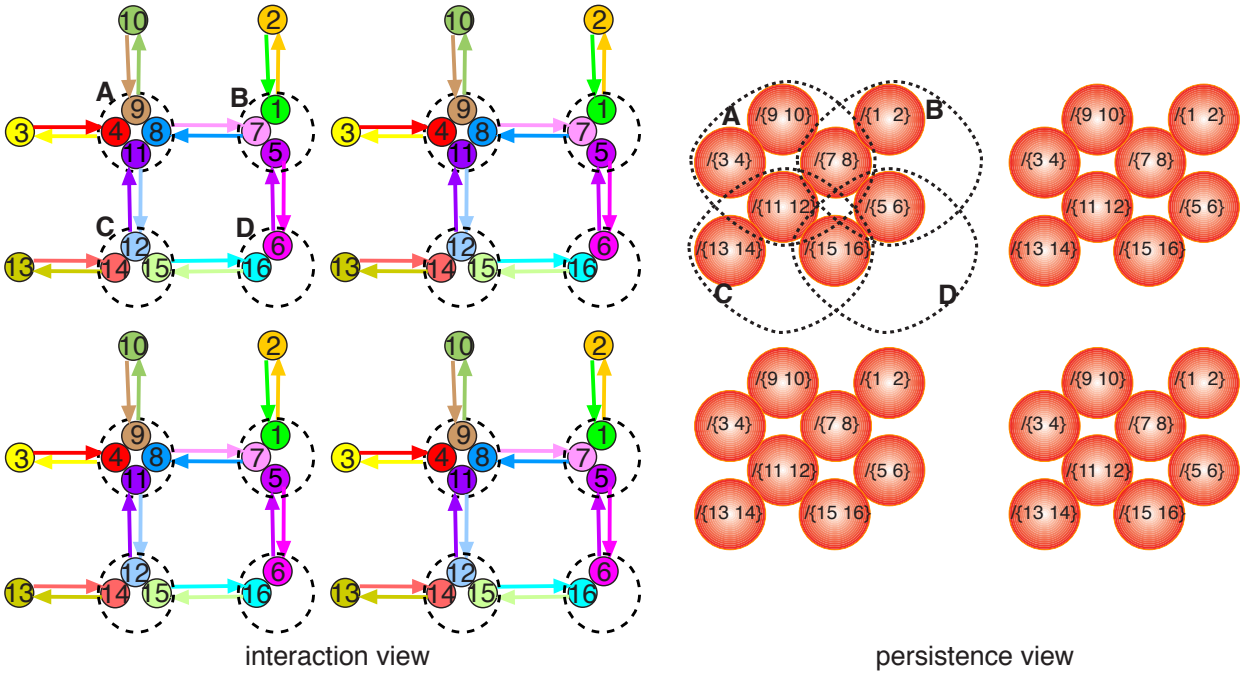


Figure 2.37. Surface tiling.

Figure 2.38 shows the tiles connected forming the tiled surface. The tiling can continue indefinitely. tile will start with a wavefront configuration. Each tiling will create a new cycle whose wavefront will be determined by the wavefront orientation of the tile. The new cycle will have opposing wavefront flow.

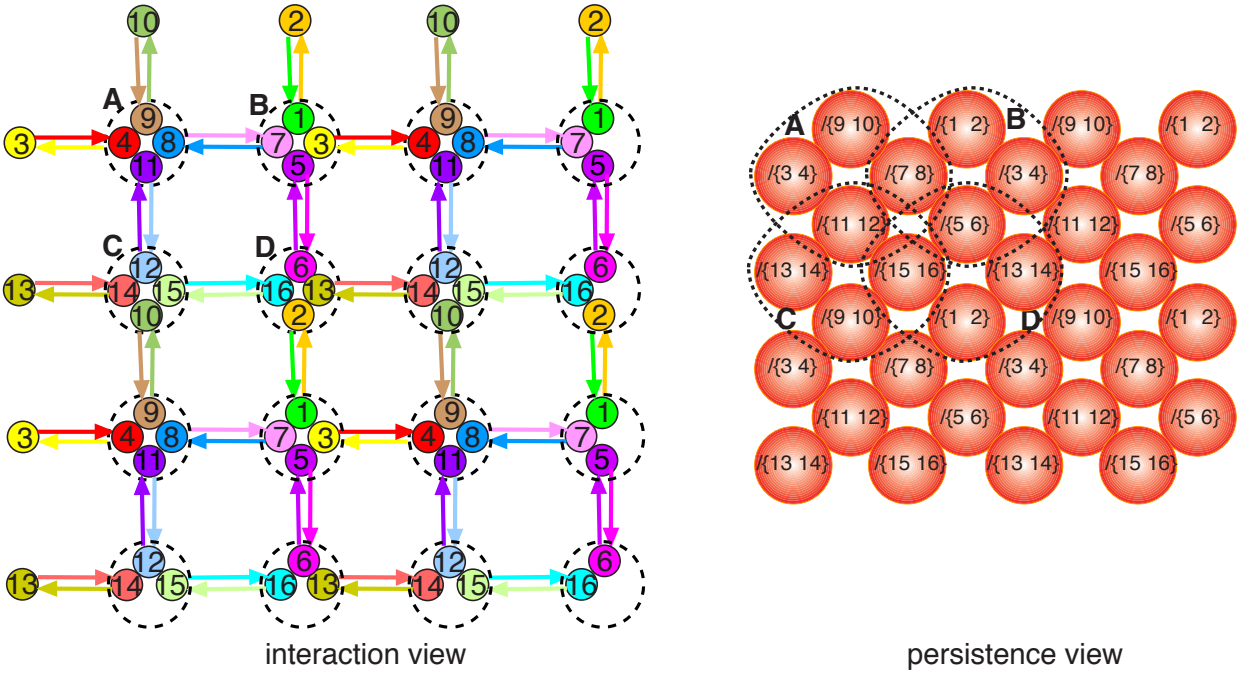


Figure 2.38. tiled surface.

An extended conjoinment of persistences using the same structure of differentness conditions over and over.

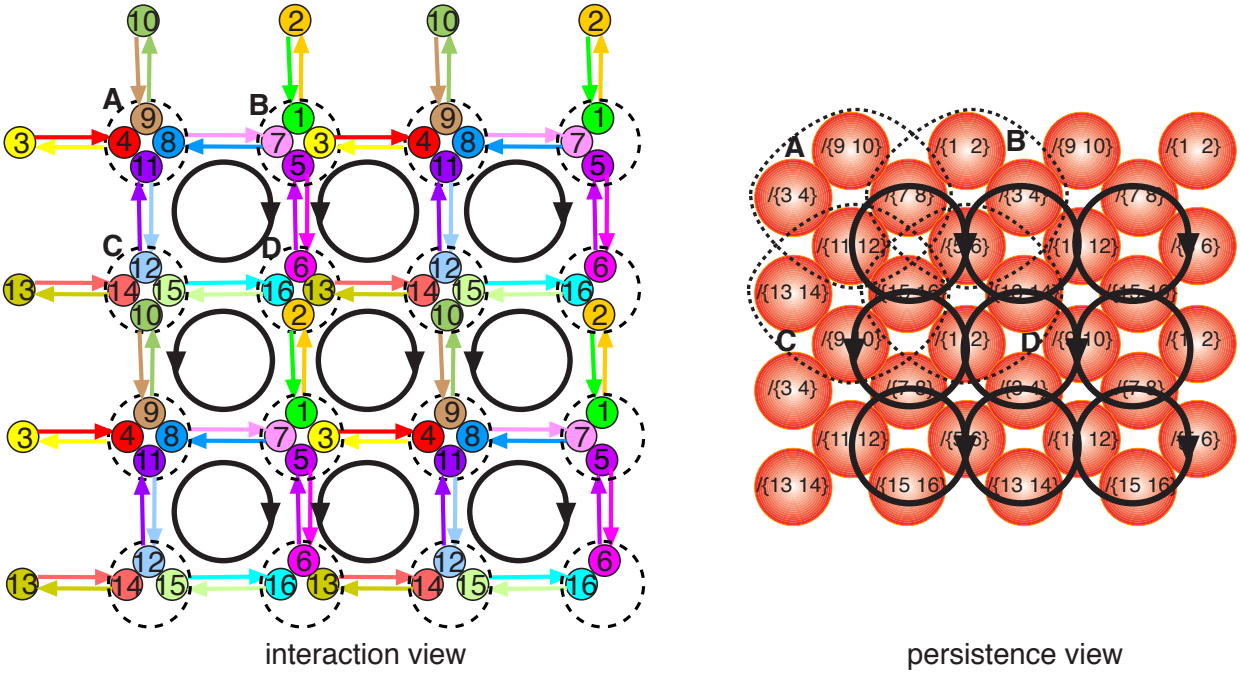


Figure 2.39. The wavefronts flowing through the conjoinment surface.

The tiles link by sharing external persistences but then behave by sharing internal persistences.

2.11.3.4. Closure

Notice that the persistences along the left edge can close with the interactions along the right edge and the persistences along the top can close with the interactions along the bottom closing the surface forming, in this case, a torus. More flexible tiling geometry, for instance with triangular conjoinments instead of quadrangular conjoinments can close the surface into a topological sphere encompassing an isolated piece of the greater condition space forming an isolated local condition space.

The wavefronts begin flowing freely within the conjoinment only with complete closure. Without complete closure the wavefronts will stall waiting for transition of dangling persistences. The surface, itself becomes a self sustaining liveness of continually cycling wavefronts, their interactions encompassing a live condition subspace. Figure 2.39 illustrates the cyclic wavefront flow in the surface conjoinment. The wavefronts in the interaction view are straightforward. The cyclic wavefront flow in the persistence view are more difficult to see. The wavefronts are not visible at all in the symbolic view.

2.12. Condition subspaces

To this point the story has been told in the context of a single condition space bounded within a gravity well, a mud puddle and so on. The next stage of extending of expressivity with differentness of isolation is the emergence of explicitly encompassed and isolated condition subspaces: shaking bags within shaking bags within shaking bags. Identical computations are isolated and differentiated in their own private condition space.

Consider in the context of Figure 2.16 that the differentness conditions **b, e, h** and **k** can penetrate the surface boundary from outside to inside and differentness condition **c, f, i** and **n** can penetrate the surface boundary from inside to outside but differentness conditions **a, d, g** and **m** are isolated within the subspace and reflect off the encompassing boundary.

A condition subspace like any other conjoinment can wander indiscriminately through the greater condition space.

2.12.1. Nature’s palettes

When an enclosed subspace managed to pinch its surface and make two closed subspaces the possibility of vast multitudes of isolated subspaces emerges each of which encloses a condition subspace that can privately host an experiment with expressions of free interactions and static conjoinments including the further encompassment of condition subspaces within that subspace. Nature’s playful palettes. Eventually the experiments move on to the conjoinment of subspaces through their encompassing surfaces.

That first condition space enclosed in its bounding surface has survived to the present. In every living cell today is a piece of the original enclosed condition space.

The point here is to illustrate that such a conjoinment that closes on itself forming a surface humming with intrinsic dynamic liveness enclosing a space of intrinsic dynamic liveness is possible with the computational resources at hand. How a lipid bilayer performs its computation magic to form an actual cell membrane and manage its permeability is another story told by other tellers.

2.12.1.1. Nature’s expressivity

Try everything without intentionality. What survives survives. The survivors are the initial learnings of nature. The survivors continue trying things in the greater condition space and accruing substantivenesses. Subspaces become loci of substantiveness that can begin taking themselves as referent, as self, in relation to the greater condition space and all the other

surviving selves. A ratchet stage of substantiveness that can begin localized ratcheting. A subspace that can begin learning in relation to itself rather than in relation to the greater condition space.

2.13. Interlude: Differentness

Computation lives on differentness.

Each persistence is a same singularity of space and of time. Each persistence is different from but not individuated from all other persistences. A persistence asserts one at a time of two or more differentness conditions. The assertable differentness conditions of a persistence are different from each other in relation to the sameness of the persistence. Differentness conditions are individuated and each differentness condition has propensity to interact with specific other differentness conditions and to not interact all with all other differentness conditions. Persistences indiscriminately encounter in condition space facilitating the interaction of their asserted differentness conditions (see section 2.1)

2.13.1. The beginning

An indefinite supply of persistences indiscriminately encountering in condition space facilitate the interaction of a limited set of asserted differentness conditions with specific interaction propensities. For instance, 92 differentness conditions asserted by 10^{28} persistences (the number of molecules in a tide pool). There are approximately 10^{26} **A**s, 10^{26} **B**s, 10^{26} **C**s and so on. Each asserted **A** is different from all of the asserted **B**s, **C**s and so on but is not different from all of the other asserted **A**s. Even though the other **A**s are asserted by different persistences the persistences are not differentiated, individuated, in condition space and do not contribute differentness to their asserted **A**s. All asserted differentness conditions **A** are the same in condition space interchangeably doing whatever **A** does in condition space.

Beginning with a huge asserted supply of a small number of differentness conditions is critical. As the number of original differentness conditions increase the probability of interaction decreases, the probability of wavefront of dependent interactions decreases and the probability of active conjoinment decreases. There is too much possibility to engender specificity. Conversely, if the quantity of original differentness condition is too small they will not be expressive enough to discover survival. Sufficiency of specificity fails from too little possibility. Presumably the atoms and molecules available in our universe are in a goldilocks range.

What differentness conditions do in condition space is specifically interact (see section 2.2). Each differentness condition has propensity to interact and change with specific other differentness conditions and to not interact with all other differentness conditions. The indiscriminately encountering persistences ensures that all possible encounters of differentness conditions occur and that all possible interactions among differentness conditions occur.

Condition space interactions are not individuated. A thousand **[A B] => [D C]** interactions is just a thousand undifferentiated interactions. This is why the blob relaxed into an indeterminate compositional mess while as a whole continuing to behave deterministically in terms of differentness conditions and their interaction propensities, i.e., as a pure condition expression (see section 2.7.9).

2.13.2. Temporal isolation

A persistence contributes to the differentness of its asserted differentness conditions through time by reasserting them one at a time over and over through its own differentnesses of time.

with each assertion of a same differentness condition isolated from the previous assertion of the same differentness condition by the assertion of one or more different differentness conditions. In the progression of differentness condition assertions A, B, C, A, B, C, A, B, C … each assertion of a same differentness condition is isolated in time from the previous assertion of the same differentness condition by the assertion of a different differentness condition. There must be at least two differentness condition alternately transitioning. A, A, A is expressionaly meaningless . The differentness of the differentness conditions contribute to their differentness through time in relation to the sameness of their asserting persistence. Only transition to a different differentness condition signifies differentness (see section 2.1.1.3).

A persistence does not contribute to the differentness of its asserted differentness conditions through condition space since the asserting persistence is always a same singularity of condition space. The persistence does not contribute any differentness of space to it asserted differentness condition.

2.13.3. Conjoinment

Persistences statically conjoin. A conjoinment of persistences as a whole inherits unique singularity of condition space from the totality of the unique spatial singularities of its component persistences. A conjoinment of persistences also inherits an isolated internal structure of “Places in relation to” from the individual spatial singularities of its component persistences. A conjoinment is a coherent ontological substantiveness, more substantive than an individual persistence which has no internal structure, that forms a subspace of determinate dependency relations within the indeterminate circumstantial relations of the greater condition space which has no places and no place relations.

2.13.4. Likelihood of interaction

When persistences statically conjoin it compromises the indiscriminate facilitation of interaction among differentness conditions in condition space. In conjoinment the asserted differentness conditions can interact only with the asserted differentness condition of directly conjoined persistences and they either interact or they do not interact. A conjoinment is either active or it is not. Where a limited quantity of primitive differentness conditions encouraged the interactions of the indeterminately encountering persistences in condition space, now the limited quantity of primitive differentness conditions encourages the possibility of the conjoined asserted differentness conditions interacting. This is the goldilocks range of primitive differentness conditions

Among multitudes of conjoinments there will be a proportion asserting persistences with propensity to interact forming active conjoinments in contrast to forming inactive conjoinments, for instance, forming rocks.

2.13.5. Isolation of conjoinment: spatial isolation

A conjoinment contributes its isolating spatial differentness to its immanent differentness conditions rendering them different from all same differentness conditions asserted outside the conjoinment. A thousand same **A** differentness conditions in a thousand conjoinments becomes a thousand differentnesses. A thousand samenesses becomes a thousand differentnesses.

2.13.5.1. Extending differentness in time and space

A conjoinment contributes to the differentness of same differentness conditions in condition space. Same primitive differentness conditions are further differentiated in time as each conjoined persistence asserts them over and over through its own differentnesses of time.

2.13.6. Isolation within conjoinment: association differentiation

As a conjoinment increases in complexity it accrues isolated persistences within its conjoinment which contribute the differentness of their isolation to the differentness of their asserted differentness conditions, i.e., they can assert, without ambiguity, the same differentness conditions as all the other persistences from which they are isolated within the sameness of the conjoinment (see section 2.8.2.1). A same differentness condition asserted by 10 conjoinment isolations extends the expressivity of the same differentness condition by 10 differentnesses with each's interaction propensities specifically limited to directly conjoined persistences. The differentiated same differentness conditions within the conjoinment are also different from all of the same differentness conditions asserted outside the conjoinment.

Isolation within conjoinment is the basis of pure association differentiation which is the expression of computation entirely within a single conjoinment (see Chapter 3.and Chapter 4). Pure association differentiation, not spontaneously constructing and evolving, is the primary expression mode of humans who provide the intentional agency of construction and purpose. The human, in turn, is a conjoinment of condition subspaces (cells) with an all encompassing internal condition subspace (blood).

2.13.7. Isolation within subspace

Conjoinments closing to encompass subspaces within condition space (See section 2.11.3) further isolates and differentiates the expressivity of differentness conditions within the sameness of the subspace. The differentness conditions and conjoinments isolated within each encompassed subspace are different by virtue of their isolation within a subspace

2.13.8. Uniqueness of isolation

The expression of isolation is by definition unique. While there can be thousands of the same differentness condition **A** assert within condition space, there can be no other expression of differentness that can claim to be a same differentness of isolation. Isolation uniquely and indefinitely extends the expressivity of differentness of the primitive differentness conditions.

2.13.9. Indefinitely extendable differentness.

The expressivity of differentness by the primitive differentness conditions is extended with differentness of isolation. A differentness condition asserted in isolation from all other same differentness conditions expresses the differentness of its isolation as well as the differentness of its differentness condition. The expressions of differentness compounds with Isolation differentness of conjoinment, isolation differentness within conjoinment and isolation differentness within subspace all further compounded with the temporal isolation and differentness of the transitioning asserted differentness condition of the persistences. The limited expressivity of differentness of the original differentness conditions is seamlessly and unambiguously extended to the indefinite expression of differentness and interaction propensity with the unlimited availability of the expression of isolation both spatial and temporal as long as there are asserting persistences. In other words the only limiting factor to the expression of differentnesses and interaction propensities is the $\mathbf{10^{28}}$ persistences available.

1. Asides comments and such will be relegated to end notes so that they do not interrupt the flow of the narrative.
   This first aside is about information. Information is generally viewed as passive. A computer manipulates information. Information guides the behavior of a computer as program, data base and so forth. This mutual dependency is one of the quandaries of computation. The differentnesses of this story might be viewed as dynamically behaving, self computing information. In other words, information manipulating itself in terms of its informationality, its expression of differentness.
2. Consider that the conventional physics view of 3D space, continuous time and deterministic causality is a point of view of particular utility. Consider further that there can be other points of view with specific topical utility. The space and time of this story is a computational point of view.

   "What physics explains right now, very deeply, are some fundamental aspects of reality but not nearly all of them." Walker, Sara Imari, *Life as No One Knows It: The physics of life's emergence*. New York: Riverhead books, 2024. p 26

   This story of computation is about those other aspects not covered by traditional physics. Physics and computation may have some intersections at certain points such as the existence of atoms and molecules but this story of computation does not appeal to any notions of conventional physics and trying to introduce such notions will detract from the story rather than contribute to the story. The physics view and the computational view are two views of space and time each entirely independent of the other for large ranges of their individual considerations. Neither explicitly references the other. Ships passing in the night.

   "I am convinced that there is entirely new physics in the living universe (the part of reality that includes living things) awaiting our discovery. ... More radically, it is not just new laws in the old paradigm that we need to dream up, but new ways of doing physics and new implications for our understanding of seemingly elementary things we think we already have a handle on like matter and time." Walker, Sara Imari , *ibid.*

   This story of computation entails new understandings of seemingly elementary things like matter in space and time

Chapter 3:
Association Differentiation

In Chapter 2 persistences associated indiscriminately and conditions interacted specifically. All expression of differentness was in terms of differentness of condition with no differentiation in terms of association of persistences. In this chapter persistences will associate specifically and conditions will interact indiscriminately (all conditions interact with each other). All expression of differentiation will be in terms of differentness of association with no differentiation in terms of differentness of condition. Interaction will be determined by direct static association of persistences rather than by interaction propensity of conditions. Each persistence will be different from all other persistences not by the differentness of the conditions it asserts but by the differentness of its place in a network of static association relations.

3.1. The behavior of statically associated persistences

An individual persistence has no associational structure, is not directional, has no input or output, no top or bottom, no right or left. Associated persistences asserting conditions with compatible interaction propensities will be continually asserting a condition to each other, each continually fulfilling an interaction propensity and continually changing its asserted condition in response.

3.1.1. Undirected continual interaction

If the conditions of associated persistences and their interaction propensities are identical as shown at the left of Figure 3.1 then the conditions of the associated persistences will interact continually and indiscriminately in all directions even with themselves.

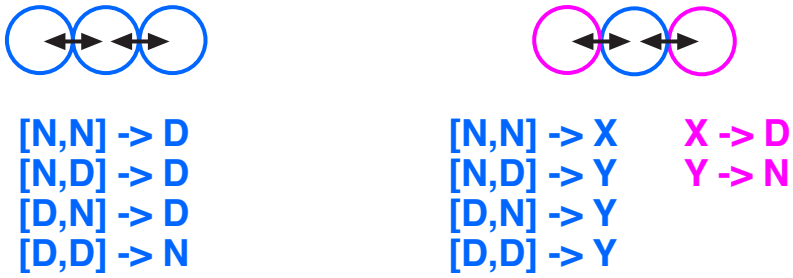


Figure 3.1. Undirected interaction of associated persistences.

A persistence can avoid interacting with its own conditions by being responsive to one set of conditions and by asserting a different set of conditions. But if it associates with a second persistence that is responsive to its asserted conditions and which asserts condition to which it is responsive, as at the right of Figure 3.1, they are dependent on each other, feed back to each other and continually interact with each other.

3.1.2. Directionalizing interaction behavior

To avoid the continual mutual interaction requires a linear association of three persistences with complementary condition interaction propensities each responsive to and asserting a different set of conditions as illustrated in Figure 3.2.

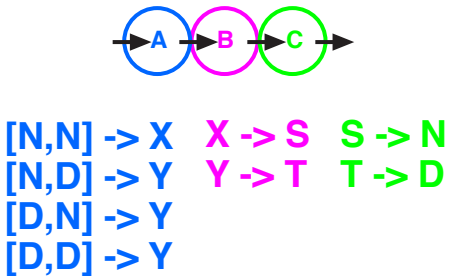


Figure 3.2. Directionalizing the behavior of statically associated persistences.

- Persistence **A** is responsive only to conditions **N** and **D** and asserts only conditions **X** and **Y**.
- Persistence **B** is responsive only to conditions **X** and **Y** and asserts only conditions **S** and **T**.
- Persistence **C** is responsive only to conditions **S** and **T** and asserts only conditions **N** and **D**.
- Persistence **C** is dependent on persistence **B**
- Persistence **B** is dependent on persistence **A** and not dependent on persistence **C**
- Persistence **A** is not dependent on persistence **B** and not dependent on persistence **C**.
- The relations of dependency become directionalized.

Conditions **N** and **D** asserted by persistence **C** do not associate to persistence **A** and do not influence the responsive behavior of persistence **A**. The **X** and **Y** conditions and the **S** and **T** conditions of persistence **B** are buffering conditions that separate the persistence **C** assertion of **N** and **D** from the responsivity of persistence **A** to **N** and **D**. It does not matter what persistence **B**'s buffering conditions are as long as they connect **A** and **C** and are different from **N** and **D**.

3.1.2.1. Differentness of place of association.

The linearly associated persistences form a **directionalized association interaction behavior** sensitive to and asserting the same set of conditions **N** and **D**. Identical input and output conditions are different because they are at different isolated places of association in relation to the dierctional interaction behavior. The directionalized association interaction behavior establishes a new form of representing interaction differentness in terms of *differentness of place of association*.

3.1.2.2. Practical directionalized interaction behaviors

A switch, Figure 3.3a, is a directionalized association interaction behavior. In Figure 3.3 the colors of the switches correspond to persistences **A**, **B** and **C**. An electromagnetic switch receives an input current condition that influences a magnetic field condition which influences a physical position condition which influences an output current condition which is isolated from the input current by interaction through two different condition domains.

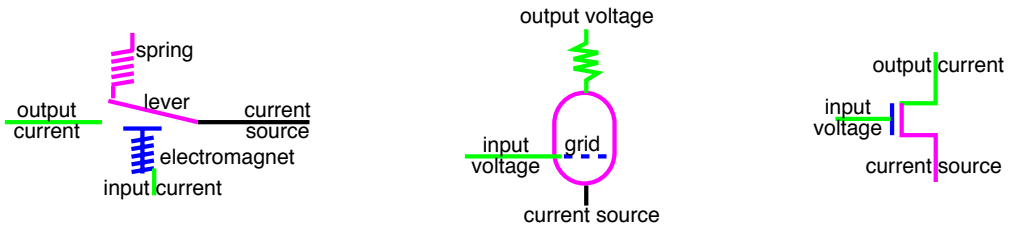


Figure 3.3. Switches isolate and directionalize condition transition flow.

An electronic tube, Figure 3.3b receives a voltage condition which influences a charge condition on the grid which influences an electron flow condition through the vacuum which influences the output voltage condition on the wire which does not feed back to and influence the input voltage condition.

A transistor, Figure 3.3c, receives a voltage condition on the gate influencing the charge condition in the channel which influences the electron flow condition through the channel which influences the voltage condition at the output which does not feed back and influence the input voltage condition.

A neuron receives input on its dendrites and asserts output on its axon which output does not feed back through the neuron to influence its dendrites

3.1.3. Differentiating interaction behavior

Directed interaction behaviors can be characterized in terms of the condition interaction propensities embodied by persistence A and the assertions of persistence C. As shown in Figure 3.4 directed interaction behaviors named **"one of"** and **"all of"** are defined in terms of the response of persistence A to condition combinations and the condition it asserts which plays through to persistence C which asserts the result condition for the interaction behavior.

The bottom of Figure 3.4 shows the unique graphic representing each type of interaction behavior with its transition table from input to output without the intermediate B interaction.

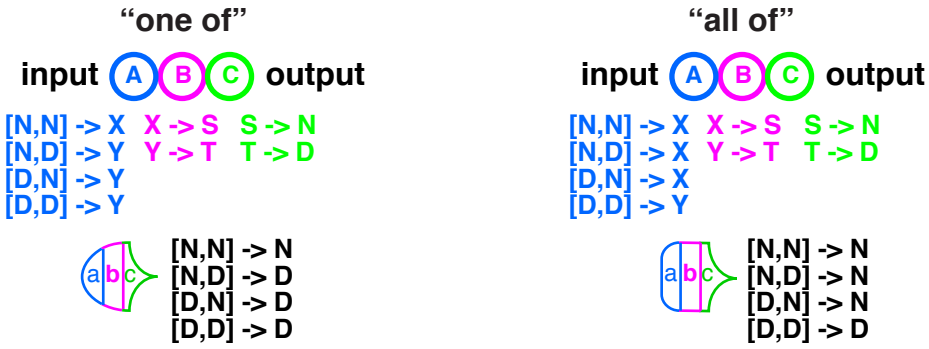


Figure 3.4. Basic interaction behaviors

3.1.4. Directed networks of directed interaction behaviors

Since every interaction behavior recognizes and asserts the same set of conditions and

isolates its output from its input, associated interaction behaviors are assured to interact enabling the indiscriminate association of interaction behaviors output to input into directed networks of dependent interaction behavior. Dependency networks of directed interaction behaviors can be represented by stretching persistence **C** to associate with persistence **A** of other behaviors as on the left of Figure 3.5. Persistence **C** can be stylistically extended to represent association as on the right.

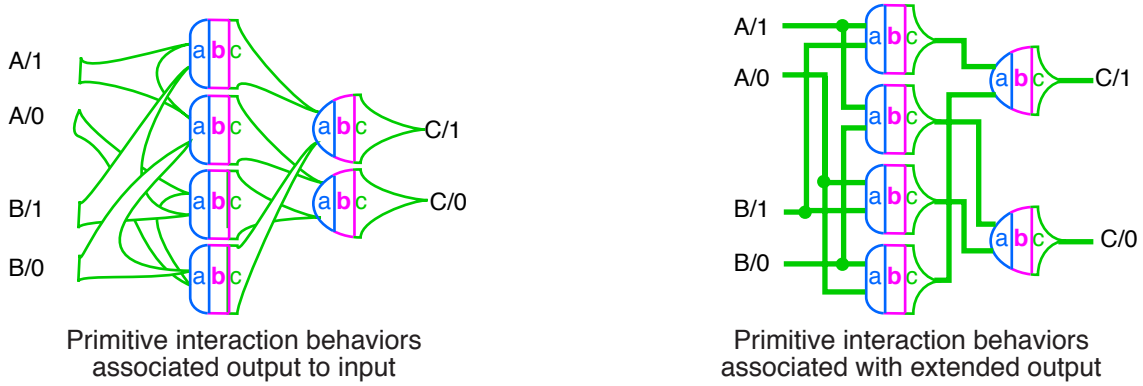


Figure 3.5. Representations of directed networks of interaction behaviors.

3.1.5. Association differentiation

- Each interaction behavior isolates its input from its output.
- The output of each interaction behavior in a network associates only to the inputs of other interaction behaviors.
- The output of each interaction behavior in a network is a uniquely different place of association in the network as shown in Figure 3.6 .
- The **D** or **N** condition asserted by each interaction behavior is different from the identical **D** or **N** condition asserted by every other interaction behavior in the network by virtue of each asserted condition being isolated at a different place of association in the network.

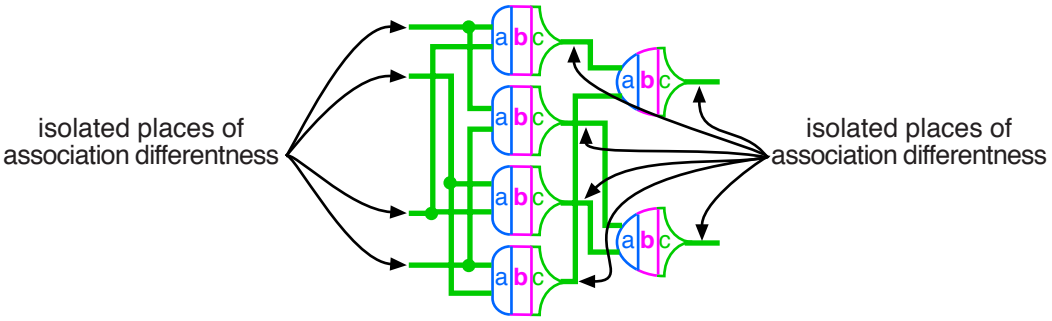


Figure 3.6. Interaction behaviors isolating different places of association.

3.1.6. Differentiating instances of interaction: expressing occurrence

The coordination of interaction is accommodated here the same way it was accommodated in chapter 2, with completeness of interactive association. In chapter 2 coordination was expressed

in relation to emptiness with the occurrence of completeness of encounter of interacting differentness conditions initiating an interaction and then the disappearance of the interacting differentness conditions and the occurrence of the interaction result condition indicating the complete fulfillment of the interaction. This appearance and disappearance of completenesses bounds and differentiates instances of primitive interaction behavior (section 2.2.4).

For an association interaction behavior, continually presented with input conditions and continually asserting its output conditions there is no disappearance of interaction conditions. No absence or emptiness separating one condition interaction from a next condition interaction, no expression of occurrence as there is in the shaking bag. This poses two difficulties.

- One: a next presentation of interaction conditions may require two or more inputs to transition. They will not transition simultaneously and the output can respond with a temporarily spurious condition to an incompletely transitioned input.
- Two: some input presentations may map to the same output condition and the transition of presented input from one interaction input to another may not cause the output to transition. If the output condition does not transition there is no intrinsic interaction behavior that differentiates instances of interaction.

These difficulties are identical to the Boolean logic difficulties of section 1.4 and, are resolved as follows.

3.1.6.1. Define an explicit representation of emptiness: non occurrence

These difficulties are addressed by assigning one condition **N** to explicitly represent "**N**ot an interaction differentness" while assigning all the other available conditions, in this case only **D**, to represent "an interaction **D**ifferentness". Condition **N** represents the absence of, emptiness of, interaction differentness. Condition **D** represents the presence of, occurrence of, interaction differentnesses. A transition from **N** to **D** represents an occurrence of interaction differentness from an absence/emptiness of interaction differentness. A transition from **D** to **N** represents a disappearance of interaction differentness into an emptiness of interaction differentness.

3.1.6.2. Define two disjoint domains of completeness

Define a representation of "completely **N**" which is all **N** assertions with no **D** assertions, i.e. completely empty of, nonoccurrence of, interaction differentness, the nonoccurrence of interaction differentness.

Define a representation of "**D** completeness" which is a pattern of **D** assertion in a background of **N** assertion. Interaction occurs with the occurrence of **D** completeness.

3.1.6.3. Require monotonic transition between completeness representations

Each interaction behavior will begin with a transition from completely **N** to **D** completeness, the appearance of completeness of interaction differentnesses. Each interaction behavior will end with, a transition from **D** completeness to completely **N**.

The monotonic transitioning between **D** completeness and completely **N** shown in Figure 3.7 unambiguously differentiating successive instances of **D** completeness is the primitive

expression of occurrence and the primitive expression of mutually exclusive differentness of computation time .

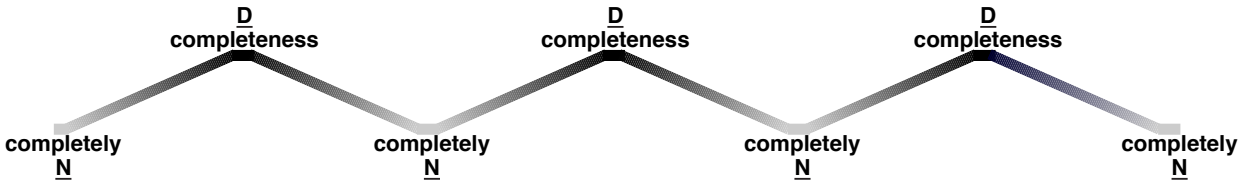


Figure 3.7. Monotonic transition of condition completeness relations.

This corresponds to the wavefront flow and counter flowing reset flow of section 2.8.4.5. For association differentiation the wavefront flow and the reset counterflow has to be standardized for all interactions.

3.2. Primitive interaction behaviors: expressing completeness

Define primitive interaction behaviors that transition their output only with presentation of input completeness fulfilling the completeness criterion (section 2.5.1). A primitive interaction behavior begins empty with inputs completely **N** and its output asserting **N**. When the inputs transition to **D** completeness the output transitions to **D** which is maintained until the input transitions to completely **N** at which point the output transitions to **N** which is maintained until the input transitions to **D** completeness at which point the output transitions to **D** which is maintained until the input transitions to completely **N** and so on.

Primitive behaviors fulfilling the completeness criterion are shown in Figure 3.8 which also shows the graphic representation and the textual representation of each interaction behavior. The "–" means that no output transition occurs. The transition of the output between **D** and **N** reflects the transition of the input between **D** completeness and completely **N**. Enclosing braces **{ }** indicate **"one of"** related behavior for which **D** completeness is one input at **D** and the rest at **N**. Enclosing brackets **[ ]** indicate **"all of"** related behavior for which **D** completeness is all inputs at **D** and none at **N**. Completely **N** is the same for all behaviors with all inputs at **N**.

The syntax <= or => indicates **"is dependent on".** Both **A<=B** and **B=>A** indicate that **A** is dependent on **B**. Figure 3.8 includes the graphic representation and the syntactic representation of each primitive interaction behavior.

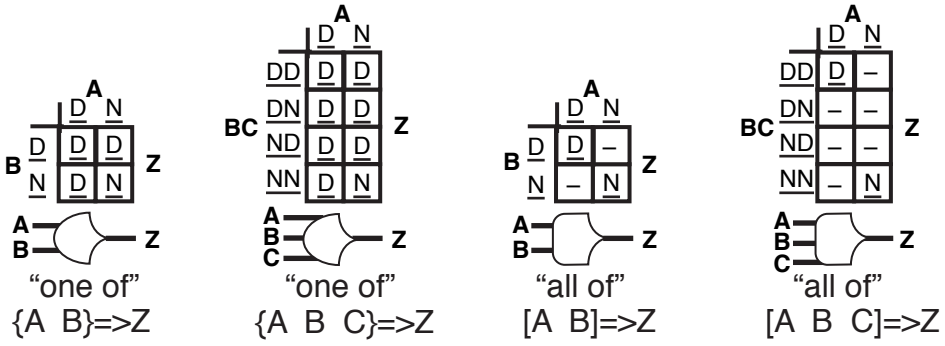


Figure 3.8. Primitive interaction behavior interaction propensities.

3.2.1. The "all of" relation, *mutual inclusivity*

Z<=[A B C]

If **"all of" A** and **B** and **C** transition from **N** to **D** then **Z** will transition to **D**. When **"all of" A** and **B** and **C** have transitioned to **N** then **Z** will transition to **N**.

The syntax structure **<=[ ]** dependently connects the different names **Z**, **A**, **B** and **C**.

3.2.2. The "one of" relation, *mutual exclusivity*

Z<={A B C}

If **"one of" A** or **B** or **C** transitions from **N** to **D** then **Z** will transition to **D**. If a second input transitions to **D** then **Z**, already at **D**, does not transition. When **"all of" A** and **B** and **C** have transitioned to **N** then **Z** will transition to **N**.

The syntax structure **<={ }** dependently connects the different names **Z**, **A**, **B** and **C**.

3.2.3. The behavior of primitive interaction behaviors

Primitive interaction behaviors are differentiated only in terms of the appreciation of their input to **D** completeness. The transition of input to completely **N** is universal for all behaviors. The behavior of the primitive interaction behaviors is illustrated in Figure 3.9.

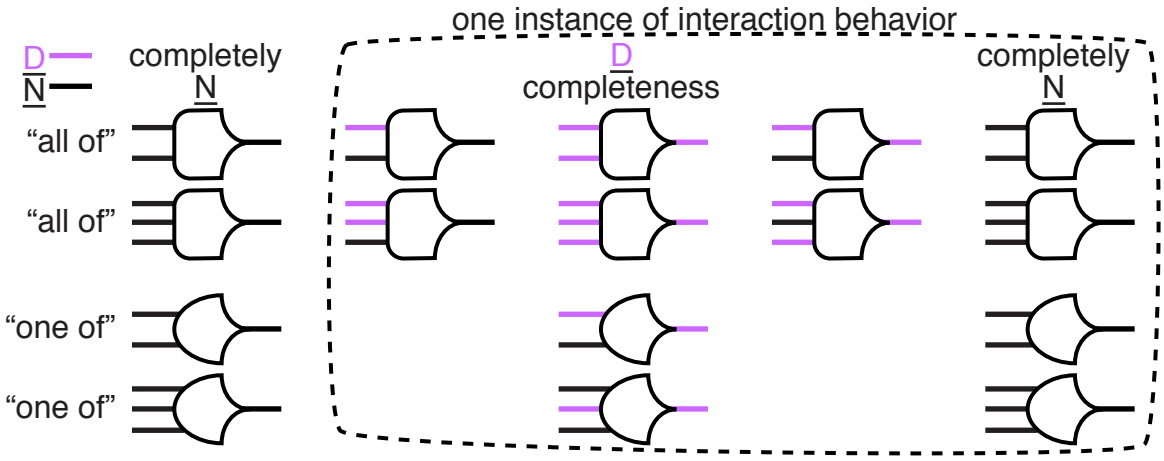


Figure 3.9. primitive behavior with monotonic completeness transition behavior.

The **"all of"** behavior is similar to the Muller C element which is generally viewed as a binary control operator in contrast to an interaction mapping behavior.* The **"one of"** behavior is identical to Boolean OR. The derivation of both elements here is quite different from their historical development.

*. David E. Muller, "Asynchronous logics and an application to information processing", Proceedings of Symposium on Applications of Switching Theory in Space Technology, H. Aiken and W. F. Main eds, Stanford University Press 1963. pp. 289-297.

3.2.3.1. The difficulties of section 3.1.6 are resolved.

- Fulfilling the completeness criterion a primitive behavior does not assert spurious output transitions due to incompletely presented input.
- Because of the monotonic transitioning between **D** completeness and completely **N** and fulfillment of the completeness criterion (section 3.3.1) every primitive behavior interaction is unambiguously accounted by the transition of its output to **D** followed by its transition to **N**.

3.2.4. Non interaction primitive behaviors

There are two primitive behaviors that do not participate in the interactions of differentnesses but that participate in the coordination of the dependent flow of interaction behavior.

The first behavior on the left of Figure 3.10 is referred to as **conversion** because it is used in network closure to convert between the two disjoint domains of completeness representation **D** and **N** (section 3.7). Conversion does not invert between two interaction differentnesses as a Boolean logic inversion does and never appears in an interaction dependency relation. The inverter symbol is still used because it is convenient and should be easily understood in context.

The single tilde ~ represents a conversion that is forced to **N** during initialization to allow an initial completely **N** wavefront to propagate through a network (Appendix D). The double tilde ~~ represents a conversion that is not forced to **N** during initialization which is used only when initializing to **D** completeness (Appendix E).

The second behavior on the right of Figure 3.10 is the **mutex/arbiter**, represented textually as **{{ }}**, that enforces two otherwise uncoordinated wavefront flows to flow mutually exclusively. In Figure 3.10 **Ain** flows to **Aout** and **Bin** flows to **Bout** but only **"one at a time"** turning two uncoordinated flows into a coherent **"one of"** related flow (Appendix C and section 4.9.3). The arbiter can be viewed as an enforcing **"one of"** behavior. The mutex/arbiter is a well understood component of asynchronous design.†

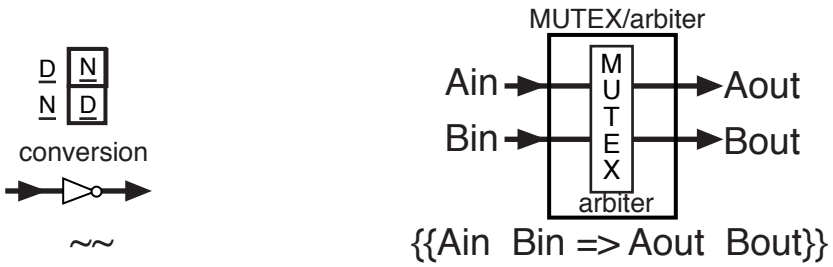


Figure 3.10. Non interacting flow coordination behaviors.

3.2.5. Initializing primitive behaviors

Figure 3.12 shows the initialization behaviors. The single tilde ~ represents a conversion that is forced to **N** during initialization to allow an initial completely **N** wavefront to propagate through a network. The less frequently occurring double tilde ~~ represents the not initialized conversion behavior which is used only with initialization to **D** completeness (see section 4.1.2).

†. Teresa Meng, "Synchronization design for digital systems", (Boston, Kluwer Academic Publishers, 1991) pp 158-163

The **“all of”** behavior can be initialized to **D** or to **N** regardless of the presented interaction input conditions. When init is enabled the result is forced to a specific condition. When init is disabled the behavior resumes its intrinsic behavior. The initialized conversion behavior is always initialized to **N**.

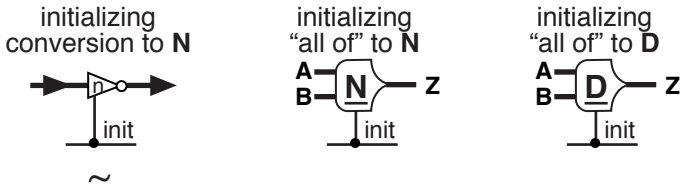


Figure 3.11. Explicitly initializable behaviors.

3.3. INTERLUDE: Sufficiently expressive primitivity‡

The association relations of persistences with spontaneously and continually interacting conditions can continue to greater complexity such as chemical molecules, macromolecules, metabolizing cells and so on but the above behaviors are sufficient to the task at hand.

Each primitive interaction behavior expresses one primitive step of mapping (interaction), condition holding behavior (memory) and input completeness appreciation (coordination). These primitive interaction behaviors along with monotonic transitioning between **D** completeness and completely **N** enable all that follows in terms of association differentiation. A static structure of differentness collaborates with a structure of dynamic differentness. It will be shown by the end of this chapter that these primitive behaviors are sufficiently expressive on their own intrinsic behavioral merits and in no need of any extrinsic assistance.

3.3.1. completeness criterion

A primitive behavior transitions its output only when its input is presented with **D** completeness or completely **N**. The transition of the output implies the completeness of the presented input and the completion of the interaction behavior.

3.3.2. Constant behavior

The primitive interaction behaviors are constant in that they always assert the same output differentness for the same presented completeness differentness allowing the primitive interaction behaviors to be indiscriminately referenced from and copied to anywhere and anywhen. In particular to compose networks of dependently related primitive behaviors.

3.3.3. The environment

A primitive behavior is entirely dependent on the presentation of input from an environment external to itself. If there is no transition of input presented there is no behavior.

‡. Readers familiar with Null Convention Logic are referred to Appendix A.

The responsible environment

If the input conditions presented by the environment to a primitive interaction behavior **monotonically transition between D completeness and completely N with an appropriate delay between transitions** then the output of the primitive interaction behavior will monotonically and correctly follow the transitioning of the input.

The environment establishes the presentation of input differentnesses and establishes the behavior interaction time. The mathematician with pencil and paper is an exemplar of a responsible environmental.

By the end of this chapter interaction networks will behave independently of the environment which will no longer have any behavioral responsibility.

3.3.4. Primitive behavior space and time

A primitive interaction behavior performs one atomic *instance of interaction* of differentness behavior between atomic places of association in one atomic *instance of interaction time* represented by the transition of a behavior's output from **N** to **D** and back to **N**. The succession of presentations of **D** completeness and completely **N** use and unuse the primitive behavior in different instances of interaction time extending the representation of differentness of the primitive behavior through time.

3.4. The dependency network: composing primitive behaviors

Primitive interaction behaviors associated output to input form a network of dependency relations among the primitive interaction behaviors.

3.4.1. Interaction dependency relations

Interaction dependency relations are expressed in terms of **mutually inclusivity** and **mutually exclusivity**. The completeness relations of mutual inclusivity [ ] (all at a time, **"all of"**) and mutual exclusivity { } (one at a time, **"one of"**) permeate interaction expression. An interaction behavior is a mutual inclusion (input completeness) of mutual exclusivities (individual input differentnesses) dependently interacting through a mapping relation to produce a mutual exclusivity (output) which proceeds on to participate in the mutual inclusivity of a next interaction.

The primitive behaviors are minimal relations of mutual inclusivity of mutual exclusivity from which larger relations of mutual inclusivity of mutual exclusivity are constructed.

3.4.2. Differentness of place of association

In a network of primitive interaction behaviors associated output to input, the output of each interaction behavior is a unique differentness of place of association within the network as illustrated in Figure 3.12 (section 3.1.5).

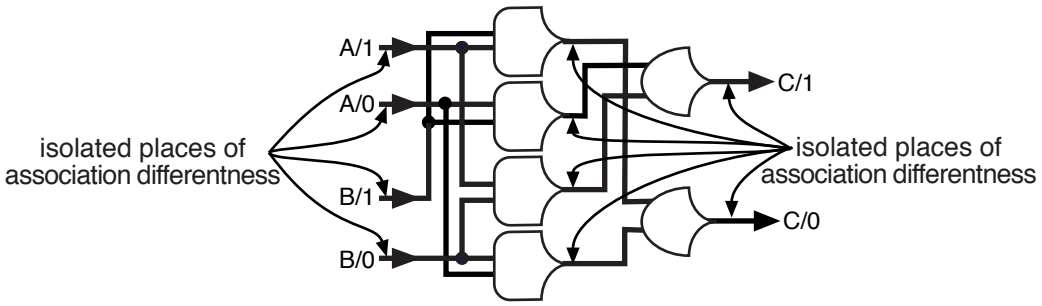


Figure 3.12. Differentness of place of association in a network of primitive interaction behaviors associated output to input.

If a primitive interaction behavior asserts condition **D** it expresses the differentness represented by its unique place of association. If a primitive interaction behavior asserts condition **N** it does not expresses the differentness represented by its unique place of association.

3.4.3. Pure association differentiation

Condition **D** is the only condition expressing interaction differentness. There is no longer any differentiation of interaction differentness in terms of condition. This is *pure association differentiation* with all interaction differentness represented by differentness of place of association and specificity of interaction represented by direct association. This is in contrast to *pure condition differentiation* (Chapter 2) in which all interaction differentness is represented by differentness of condition and its propensity to interact.

3.4.4. First level localities of interaction differentness

A first level locality is two or more places of association differentnesses associating to form a locality of mutually exclusive interaction condition differentness with **"one of"** its places of association transitioning to **D** while the other places remain at **N** (**D** completeness). A locality is empty of differentness when all of its places are asserting **N** (completely **N**). **A**, **B** and **C** in Figure 3.12 are localities of mutually exclusive interaction condition differentness. A network of associated primitive behaviors is also a network of interacting localities.

Multi-rail and particularly dual-rail encoding has long been understood as a delay insensitive encoding for differentness (information) transfer with "**N**" typically referred to as a spacer state.§ It has been less well understood as a fundamental aspect of interaction differentness.

The effective differentness a locality expresses is a composition of two differentnesses:

- the interaction condition differentness it asserts with **D** completeness
- the differentness of its unique place of association within the network

3.4.4.1. Expressing a first level locality

Differentnesses of a locality are expressed in terms of, "**one of"**, and, **"all of"**, withiness relations. The locality named **C** of Figure 3.12 which can assert **"one of"** two places of

§. Tom Verhoeff, "Delay Insensitive Codes–An Overview", Distributed Computing (1988) 3, Springer Verlag, pp:1-8

association named **1** and **0** each of which can assert **“one of”** **D** or **N** is represented as two **“one of”** relations expressed as:

C/{1 0}/{D N}

The slash / indicates a withiness relation. The braces **{ }** indicate **“one of”** related. The expression can be read as **C** is dependent on **“one of” 1** or **0** each of which is dependent on **“one of” D** or **N**. The above expression expands to:

C/{1/{D N} 0/{D N}}

The three localities **A**, **B** and **C** of Figure 3.12 are expressed as

A/{1 0}/{D N} B/{1 0}/{D N} C/{1 0}/{D N}

Since every place of association can only assert **D** or **N** the **{D N}** term is a universal most primitive terminal and can be implied with a terminal dangling slash.

A/{1 0}/ B/{1 0}/ C/{1 0}/

Which expands to:

A/{1 0}/{D N} B/{1 0}/{D N} C/{1 0}/{D N}

A, **B** and **C** which all have the same locality structure can also be represented as:

(A B C)/{1 0}/

Which distributively expands to

A/{1 0}/ B/{1 0}/ C/{1 0}/

and so on.

Each reference is a pathname to the place of association which mirrors the place’s assertion of **D** or **N**:

A/1 A/0 B/1 B/0

and so on.

A locality as a whole is referenced with the locality name which mirrors the behavior of the locality as a whole.

A B

E/{3 2 1 0}/

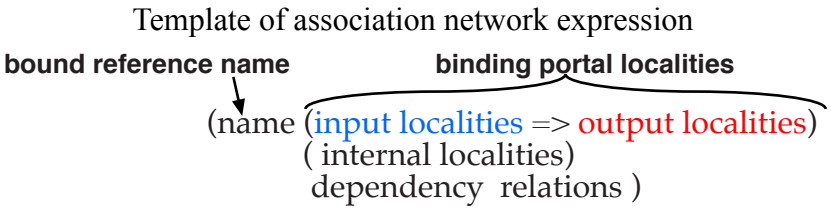
Expresses a locality named **E** with four mutually exclusive differentnesses named **3**, **2**, **1** and **0** represented as four places of association only one of which will assert **D** at a time.

Second level localities with larger ranges of mutually exclusive interaction differentness are composed by associating first level localities with small ranges of interaction differentness (section 3.4.9.).

A locality mirrors a persistence of pure condition differentiation of Chapter 2 as a locus of interaction transition and bearer of mutually exclusive differentness.

3.4.5. Expressing a network of dependency relations

A network expression, enclosed in parentheses, includes an expression *reference name*, a *binding portal* specifying the localities exposed to the *external environment* with their input to output dependency relation, the specification of referenced *internal localities* and the *dependency relations* among the localities and behaviors of the network.



In Figure 3.12 locality **A** expressing a mutually exclusive differentness and locality **B** expressing a mutually exclusive differentness mutually inclusively associate to interact and produce the mutually exclusive differentness of locality **C**. The expression below completely expresses the network. To keep directionality conveniently visible input localities are shown in **blue** and output localities are shown in **red**. There are no internal localities in this expression. In Figure 3.13 the primitive behaviors **"one of"** and **"all of"** are referenced respectively with **{ }** and **[ ]** encompassing a list of input association places composed with reference nesting relations as with:

{[A/0 B/1] [A/1 B/0]}

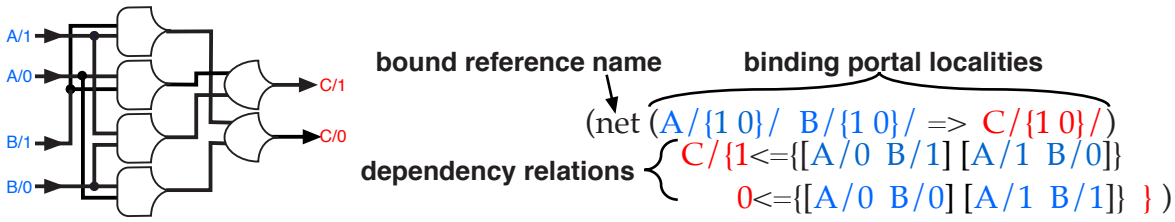


Figure 3.13. Network expression for Figure 3.12

The expression reference name is **net**. The binding portal expresses that output locality **C** is dependent on, **=>**, the input localities **A** and **B**. The specific dependency relations of the network are expressed by recapitulating the expression of output assertion locality **C** and nesting within the recapitulated expression the dependency relation, **<=**, for each component of the locality, **{1 0}**, on components of input localities **A** and **B** in terms of the primitive behaviors **"all of" [ ]** and **"one of" { }**. For instance, [**A/0 B/1**] represents an **"all of"** behavior dependent on inputs **A/0**

and **B/1**. If both **A/0** and **B/1** assert **D** then the **"all of"** relation asserts **D**. **{[A/0 B/1] [A/1 B/0]}** represents a **"one of"** behavior dependent on **[A/0 B/1]** and **[A/1 B/0].** If **"one of"** the relations **[A/0 B/1]** or **[A/1 B/0]** asserts **D** then the **"one of"** relation **{[A/0 B/1] [A/1 B/0]}** will assert **D**.

If the term **{[A/0 B/1] [A/1 B/0]}** resolves to **D** then **C/1** transitions to **D**. If the term **{[A/0 B/0] [A/1 B/1]}** resolves to **D** then **C/0** transitions to **D**. Only one of the terms will resolve to **D**. The mutual exclusivity of the terms traces back to the mutually exclusivity of the differentnesses of locality **A**, the mutual exclusivity of the differentnesses of locality **B** and their cross association (section 3.4.8).

The expression of the localities and their dependency relations maps directly to the network of Figure 3.12.

3.4.6. The binding portal

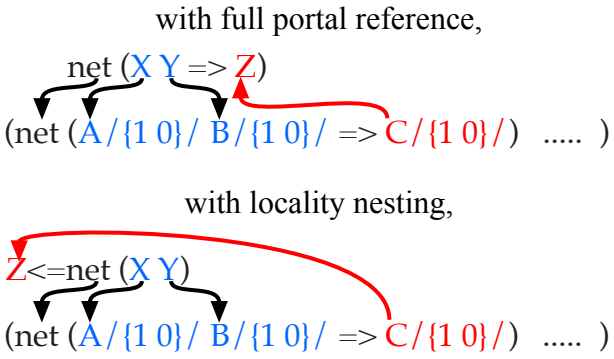
There are three forms of dependency association relation. The binding portal serves as an association hub expressing all three association relations.

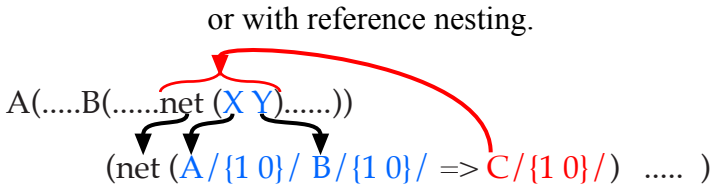
1. One to one relations
   The binding portal allows the network to be referenced externally and to associationally bind each portal locality with differently named external locality.
2. Many to one relations
   A binding portal typically maps multiple inputs to a single output.
3. One to many relations
   The binding portal associates each input locality by name correspondence to multiple places of association within the dependency expression.

3.4.6.1. One to one reference association through the exposed binding portal

A network expression is exposed to reference by an *external environment* through its binding portal in terms of its expression reference name and the syntax structure of its binding portal. The only name within a network expression that is externally visible is the expression reference name. All other names within the expression can only be referenced from within the expression. The same names can be reused outside the expression without ambiguity.

External locality names are bound to internal locality names by corresponding syntax structure within the binding portal. There are three ways of referencing a network expression through its binding portal:



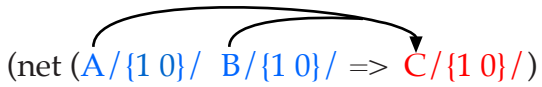


With full portal reference and locality nesting the output locality named **C** is referenced by the external locality named **Z** which can be further referenced distributing the differentness of locality **C**. reference nesting is an unnamed one to one relation which does not support further reference to and distribution of the differentness of locality **C**.

All of these variations of reference are used in the examples.

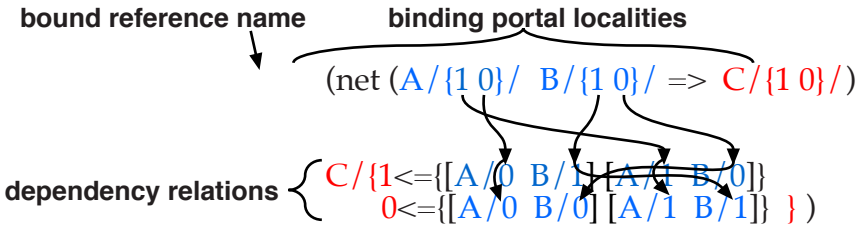
3.4.6.2. Many to one association in the binding portal

The many to one association of input to output in the binding portal.



3.4.6.3. One to many association into the dependency relations of a network

The binding portal distributes the differentness of each of its localities by name correspondence association into the expression of dependency relations.



3.4.6.4. The external environment

A binding portal exposes a network to a presentation from an uncharacterized environment beyond its binding portal on which presentation the network is completely dependent for its liveness (no presentation no network behavior), its temporal reference (with the monotonic transitions between **D** completeness and completely **N**) and to provide variability of behavior (The network itself is constant. Its behavior varies only with the differentness of its presented input). The external environment is in complete control of the network through its binding portal.

3.4.6.5. Network wholeness

The binding portal bounds the network with input and output and defines its wholeness of interaction behavior. This wholeness of network interaction behavior is not a wholeness of expression which extends through the binding portal into the environment which is not accounted by the network.

3.4.7. Wavefronts of transition

A transition wavefront consists of behaviors transitioning their output in response to transitioning of presented input which transitioned outputs are presented to the input of

subsequent behaviors which transition their output and so on. Flowing wavefronts of transition shuttle the weft of transition differentness through the warp of network differentness weaving the fabric of computation.

An interaction begins with a transition to completeness of presentation of one or more interacting differentnesses to the input of the exposed binding portal of a constant network which initiates a wavefront of transition flowing through the network to its binding portal output. This wavefront of transition flow from the completeness of presented input to the completeness of output must be unambiguous for all possible input presentations and all possible internal delay relations.

3.4.7.1. The unambiguous wavefront of transition to D completeness

Figure 3.14 illustrates a transition to **D** completeness wavefront flowing through the network of Figure 3.12 followed by the transition to completely **N** wavefront. The network begins empty of interaction differentness with all localities completely at **N,** Figure 3.14a. The external environment transitions the input localities from completely **N** to **D** completeness. **A/0** transitions to **D** forming **D** completeness for locality **A**, Figure 3.14b, then **B/1** transitions to **D** forming **D** completeness for locality **B**, Figure 3.14c, forming network input **D** completeness. The inputs are held at **D** completeness as a wavefront of transitions from **N** to **D** begins flowing through the network. Because the inputs are held at **D** completeness, because of the completeness behavior of the primitive interaction behaviors and because there are only transitions from **N** to **D** with no transitions from **D** to **N** there are only correct transitions flowing through the network. There are no incorrect or spurious transitions: no glitching.

Due to the cross association, the output assertion of the network **C** transitioning to **D** completeness**,** Figure 3.14d, implies that all of the input has transition to **D** completeness, that the consequent transition to **D** wavefront has propagated through the network and that the transitioned output is the correct resolution of the presented input as well as the necessarily last transition to completeness of the wavefront flow through the network indicating completeness of interaction (section 3.4.8.2). The transition of the output to **D** completeness corresponds to the **Z** condition of the Roman Numeral example of Chapter 2

3.4.7.2. The unambiguous wavefront of transition to completely N

The transition to **D** completeness wavefront is followed by the external environment presenting a transition to completely **N** to the input. **A/0** transitions to **N** forming completely **N** for locality **A**, Figure 3.14e, then **B/1** transitions to **N** forming completely **N** for locality **B**, Figure 3.14f forming completely **N** for the network input. The inputs are held at completely **N** as a wavefront of transitions from **D** to **N** begins flowing through the network. Again, because the inputs are held at **N**, because of the completeness behavior of the primitive interaction behaviors and because there are only transitions from **D** to **N** with no transitions from **N** to **D**, there are only correct transitions flowing through the network. There are no incorrect or spurious transitions: no glitching.

Again, due to the cross association, when the output of the network **C** transitions to completely **N**, Figure 3.14g, it implies: that the input transition to **N** wavefront is complete, that the transition to **N** wavefront has propagated through the network and that the network is

completely empty of interaction differentness (excepting orphans, see section 3.4.8.1 and 3.4.8.2).

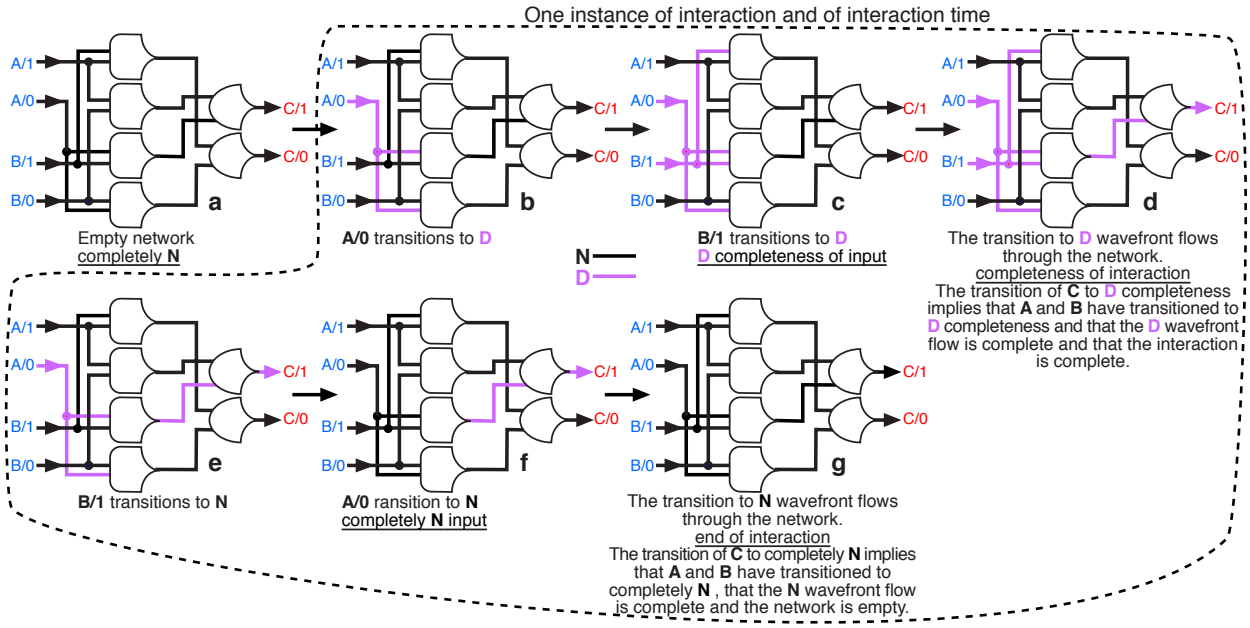


Figure 3.14. Flow of transition wavefronts through the network.

3.4.7.3. Singular appreciability

The network output locality transition to completeness is the only singularly appreciable event of a network interaction marking one instance of interaction and one instance of interaction time. It is the network counterpart of a temporal instant. Transition events within the network occurring with differing delays are not singularly appreciable but are temporally incoherent in the context of the network as a whole. It is the completeness behavior that coordinates the flow behavior of the network not any synchronization of timing behavior.

3.4.7.4. Network ephemeral instances of interaction

A network can perform only one interaction *at a time*. All transition behavior in the network is subordinate to and dependent on presentation to the binding portal input and its consequent wavefront flow through the network. A transition to **D** completeness wavefront flowing through the network from its binding portal input presentation to its binding portal output transition to **D** completeness and flowing out of the network into the external environment is one **instance of interaction** for the network. Interaction result wavefronts don't necessarily end in the external environment. They are just unaccounted by the network after they flow out.

The following transition to completely **N** wavefront flowing through the network and transitioning the output to completely **N** empties the network of interaction differentness retaining no record, no history, of the instance of interaction.

With each transition to **D** completeness a network is used. With each transition to completely **N** the network is unused. With the next transition to **D** completeness the network is reused

extending the expressivity of the network through time. Each instance of interaction occurs and then irretrievably disappears from the network.

3.4.8. Recognizing presented input

To assert an output based on the differentness of its presented input a network must first appreciate the presented input differentness. The input to the network forms a locality of mutually exclusive differentness composed of the individual input localities. The mutually exclusive differentnesses for locality A/{1 0}/ are named **1** and **0**. The mutually exclusive differentnesses for locality B/{1 0}/ are named **1** and **0**. The two inputs cross associate to form a composite range of mutually exclusive differentnesses named, **00 01 10** and **11**, only one of which is presented at a time by the two input localities.

"one of" A/1 or A/0 will assert **D** and **"one of"** B/1 or B/0 will assert **D**. Only one of the cross associations will present **DD** which is appreciated by one of a rank of **"all of"** behaviors searching for the one fulfilling cross association. The one appreciating **"all of"** behavior, recognizing the presented input differentness, will transition its output to **D** determining whether C/1 or C/0 transitions to **D**.

The output of the rank of **"all of"** behaviors is itself a locality of mutually exclusive differentness which can be named and referenced explicitly expressing the cross association search as in Figure 3.15 instead of it being implied in the nesting relations of the network dependency expression as in the expression of section 3.4.5.

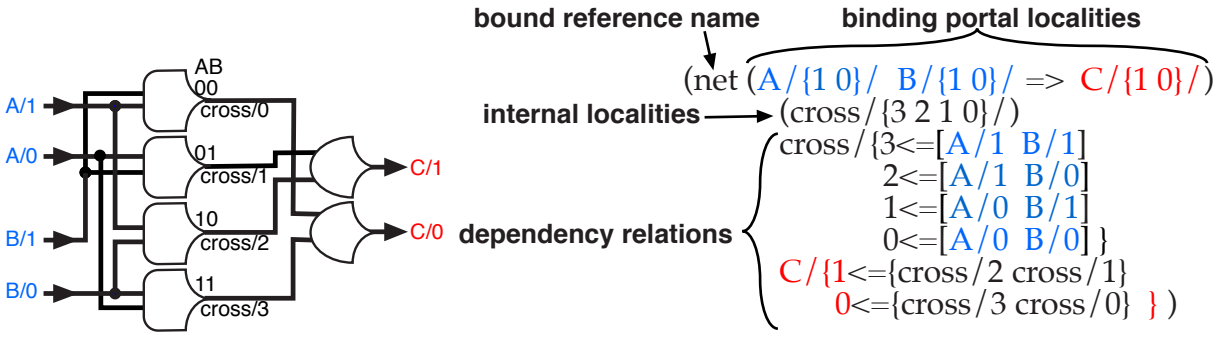


Figure 3.15. The network expression with explicitly named cross association locality

3.4.8.1. Search failures - The orphans

As the transition to **D** wavefront propagates through a cross association search there are ineffective search branches that fail to form cross association completeness which are called orphans because they have lost their relations. In Figure 3.16a the effective search branches of the transition to **D** completeness wavefront are highlighted in purple and the ineffective orphan branches of the transition to **D** completeness wavefront are highlighted in green.

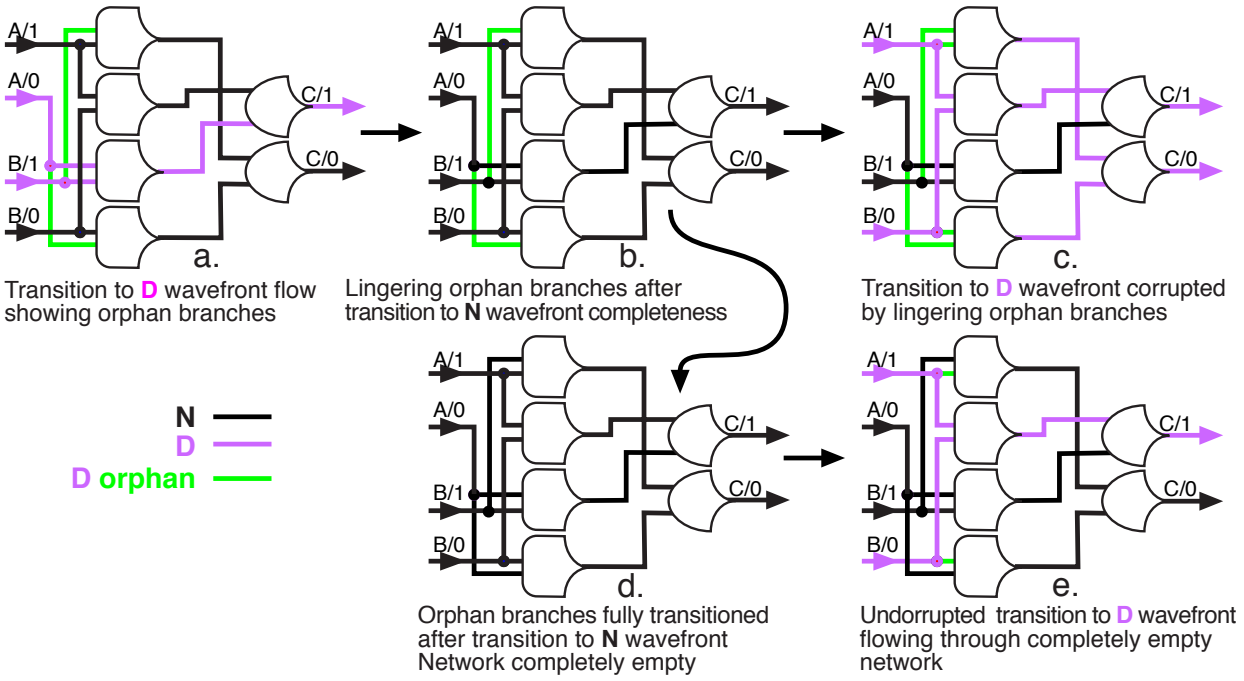


Figure 3.16. The orphans.

When the inputs presenting **D** completeness transition to completely **N** a wavefront of transition to completely **N** propagates through the network. When C transitions to completely **N** it implies that the input presentations have transitioned to completely **N** and that the effective search branches of the transition to **D** wavefront have transitioned to **N** but it does not imply that the ineffective orphan branches have transitioned to **N** which they may not have as in Figure 3.16b.

There must be an assumption that every orphan branch will transition to **N** before the next transition to **D** wavefront arrives at the branch. If the next transition to **D** wavefront arrives before the orphan branches transition to **N** the search is corrupted as in Figure 3.16.c. The differentnesses of locality C should never both be **D**. If the orphans transition to **N** before the next transition to **D** wavefront arrives as in Figure 3.16d the next transition to **D** wavefront is not corrupted and the result will be as in Figure 3.16 e.

It is shown in Appendix G, that if orphan paths are isolated to a single branch and do not propagate through a primitive interaction behavior then orphan branches will always transition to **N** well before the next transition to **D** wavefront can arrive at the branch.

The orphan is similar to the Martin notion of the isochronic fork which has a more stringent relative delay requirement because it relates to undefined external relationships of which worst case behavior must be assumed. The orphan delay sensitivity is less stringent because it relates well defined internal relationships.

Notice that in the pure condition expression of Chapter 2 search failures to form interactive completeness do not have lingering implications. A persistence and its asserted condition simply do nothing and continue searching (Section 3.4.8.2).

3.4.8.2. The completeness criterion

The completeness of the output must imply the completeness of all of the presented input. The primitive behaviors fulfill the completeness criterion but a constant network does not fulfill the completeness criterion by virtue of its composition with completeness fulfilling primitive behaviors. There can be networks that do not fulfill the completeness criterion.

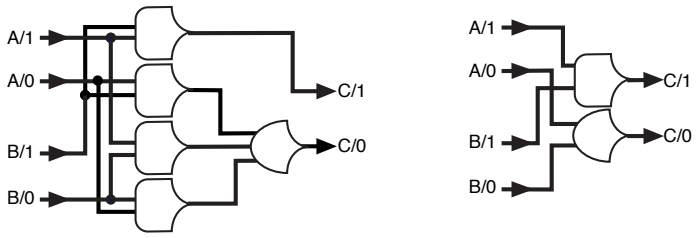


Figure 3.17. Fulfilling and not fulfilling the completeness criterion

The network on the left of Figure 3.17 fulfills the completeness criterion. The network on the right performs the same interaction mapping behavior asserting the correct result but it does not fulfill the completeness criterion. The output **C/0** can transition to **D** with just one input transitioned to **D** completeness.

The cross association not only recognizes the presented input but demands the completeness of the presented input. This demand for completeness of input ensures that the asserted output of the network will not transition until the input is completely presented. The singular transition of the network output to completeness indicates that the input of the network has transitioned to completeness, that the consequent wavefront has propagated completely through the network and that the asserted output is the correct result for the presented input for both the transition to **D** completeness and the transition to completely **N**.

3.4.9. Second level localities of interaction differentness

The cross association search in which two input localities each with a small range of mutually exclusive differentnesses mutually inclusively combine to form a single locality with a larger range of mutually exclusive differentness (see Figure 3.15) inspires the means of expressing localities with large ranges of mutually exclusive differentness in terms of mutually inclusive combinations of localities with small ranges of mutually exclusive differentness.

The second level locality expressed as:

S/[D C B A]/{1 0]/

which contains **“all of”** the second level differentness names **D**, **C**, **B** and **A** each associated with the first level differentness names **0** and **1** representing places of association each of which can assert **D** or **N** expands to:

S/[D/{1 0]/ C/{1 0]/ B/{1 0]/ A/{1 0]/]

which expands to:

S/[{D/1 D/0]/ {C/1 C/0]/ {B/1 B/0]/ {A/1 A/0]/]

The mutually inclusive relation means that there are now multiple places of association expressing the differentnesses of the locality and a reference now consists of multiple pathnames to the multiple places of association.

A reference to **S** includes "**all of" "one of" D/1** or **D/0**, **"one of" C/1** or **C/0**, **"one of" B/1** or **B/0** and **"one of" A/1** or **A/0.** A differentness, a **D** completeness, of the locality is referenced as:

S/[D/1 C/0 B/0 A/1]

The first level differentness names are self bounding single symbols they can be implicitly associated with second level differentness name by order of occurrence in the second level expression so the above reference can become

S/[1001]

The "all of" reference can be implied from the locality expression

S/1001

A place value number, for instance, can be expressed as a second level locality. A locality named **two bit** with a range of 4 mutually exclusive differentnesses is expressed as:

twobit/[B A]/{1 0}/ which expands to

twobit/[B/{1 0}/ A/{1 0}/] which expands to

twobit/[{B/1 B/0} {A/{1 A/{0}] and so on

is referenced as

twobit/00 twobit/01 twobit/10 twobit/11

Bits are referenced as

twobit/B twobit/A

The second level locality as a whole is referenced as

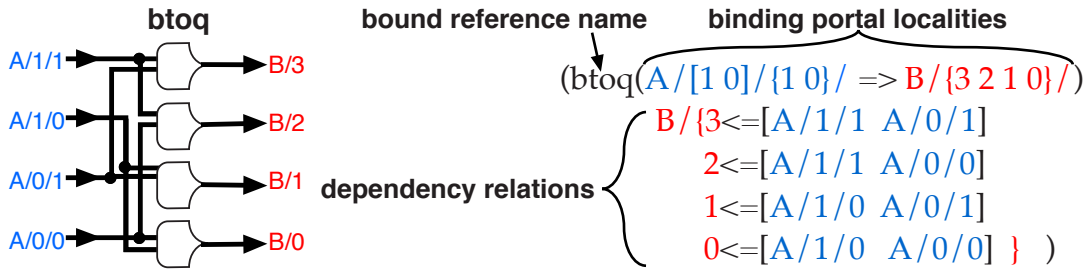
twobit

and so on.

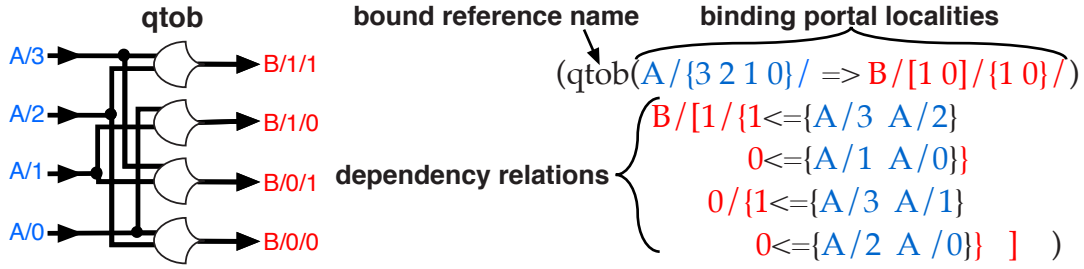
3.4.10. Composing larger constant networks

Larger constant networks are composed from smaller constant networks by associating binding portal output to binding portal input. The first example is a composition of two component constant networks. Network **btoq** converts a two digit binary radix 2 representation

to a one digit quaternary radix 4 representation. Network **qtob** converts quaternary back to binary.



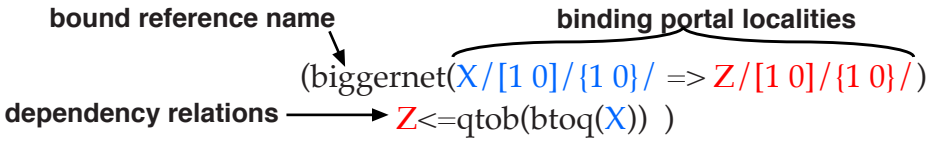
The components of locality **B/[3 2 1 0]** are dependent on relations among components of input locality **A[1 0]/{1 0}**.



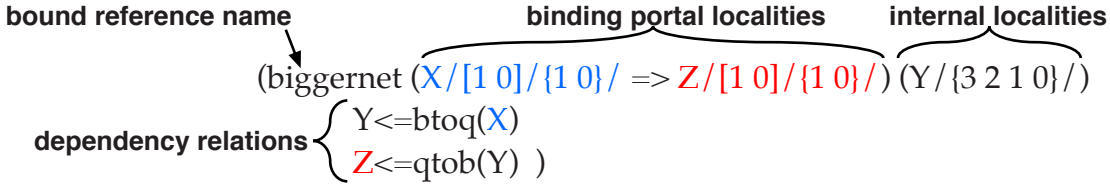
The components of locality **B/[1 0]/{1 0}**, are dependent on relations among components of input locality **A[3 2 1 0]**.

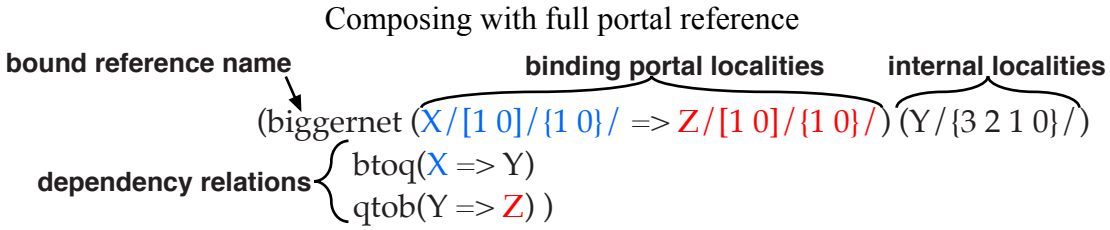
The two component constant networks are composed into a larger constant network named **biggernet** by representing the dependency relation between binding portals with reference nesting, locality nesting or full portal reference.

Composing with reference nesting and locality nesting



Composing with locality nesting





Locality **Z** as a whole is dependent on the asserted differentness of **qtob** which is dependent on the asserted differentness of **btoq** which is dependent on the presented differentness of **X** as a whole. The intermediate locality **Y** is also dependent as a whole. All three of the above dependency expressions specify the network of Figure 3.18.

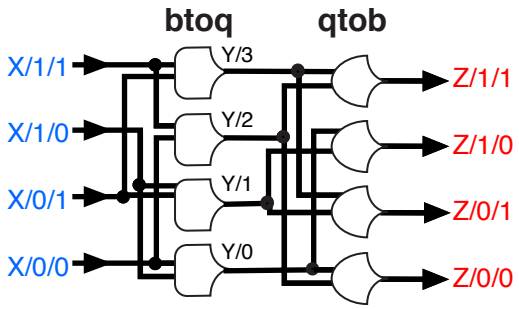


Figure 3.18. Composed constant network.

3.4.10.1. Associating exposed binding portals

When output portion of a binding portal is associated an input portion of a binding portals the portions previously exposed to the uncharacterized external environments (section 3.4.6.4) just beyond the exposed binding portals become incorporated into and fully characterized within the composite network. The unassociated portions of the binding portals form the exposed binding portal of the new composite network as a whole which retains the singular appreciability of its interaction as a whole. The newly component network becomes subordinate to and dependent on the new exposed binding portal and its input wavefront of transition for their liveness and time (section 3.4.10.8).

No matter how big and complex a composed network becomes it never outgrows the need for an exposed binding portal to an uncharacterized external environment in complete control of its behavior (section 3.4.6.4).

3.4.10.2. Locality correspondence

When an output locality of one network is associated to an input locality of another network the structure of the associated localities must correspond exactly in that they effectively express a same single locality, just with different names. The structure of locality **Z** must match the structure of locality **qtob/B**. The structure of locality **X** must match the structure of locality **btoq/A**. The input locality structure of locality **qtob/A** must match the corresponding nested output locality structure of **btoq/B** to which the intermediate locality **Y** must also correspond.

3.4.10.3. Inheritance of locality structure

One approach to expressing locality structure correspondence is to require that all locality expressions be fully specified at all levels and that the structure of all dependently related localities correspond. Another approach is to allow a locality to be referenced by name only and to inherit its structure from an already fully expressed reference to the common locality. There can be multiple references to the same locality and they must all correspond. If some references are partial they can be filled in from other references. The references among themselves must compose to a single expression of the locality. The **biggernet** dependency expression above can be rendered as:

(biggernet (X -> Z)
Z<=qtob(btoq(X)))

Z inherits locality structure from **qtob/B/[1 0]/{1 0}/** resulting in

Z/[1 0]/{1 0}/

X inherits its locality structure from **btoq/A/[1 0]/{1 0}/** resulting in

X/[1 0]/{1 0}/

biggernet expanded with the inheritance is

(biggernet (X/[1 0]/{1 0}/ -> Z/[1 0]/{1 0}/)
Z<=qtob(btoq(X)))

The **biggernet** dependency expression can also be rendered as:

(biggernet (X -> Z) (Y)
Y<=btoq(X)
Z<=qtob(Y))

with the inheritance including **Y** inheriting from **btoq/B** and from **qtob/A**:

(biggernet (X/[1 0]/{1 0}/ -> Z/[1 0]/{1 0}/) (Y/{3 2 1 0}/)
Y<=btoq(X)
Z<=qtob(Y))

Inheritance simplifies expression and reduces opportunities for errors.

3.4.10.4. Composing constancy and completeness behavior

A network is constant if it always asserts the same result differentness for the same interacting input differentnesses. If each component network of a composite network is constant (section 3.4) and fulfills the completeness criterion (section 3.4.8.2) then the composite network as a whole will be constant and fulfill the completeness criterion. The point is illustrated in Figure 3.19 showing a larger network composed with the network of Figure 3.12 copied three times.

The component network expression is:

(net(A/{1 0}/ B/{1 0}/ => C/{1 0}/)
C/{1<=={[A/0 B/1] [A/1 B/0]}
0<=={[A/0 B/0] [A/1 B/1]} })

The composed network expression is:

(bignet(W X Y Z => C)
C<=net(net(W X) net(Y Z)))

Which expands with inheritance to:

(bignet(W/{1 0}/ X/{1 0}/ Y/{1 0}/ Z/{1 0}/ => C/{1 0}/)
C<=net(net(W X) net(Y Z)))

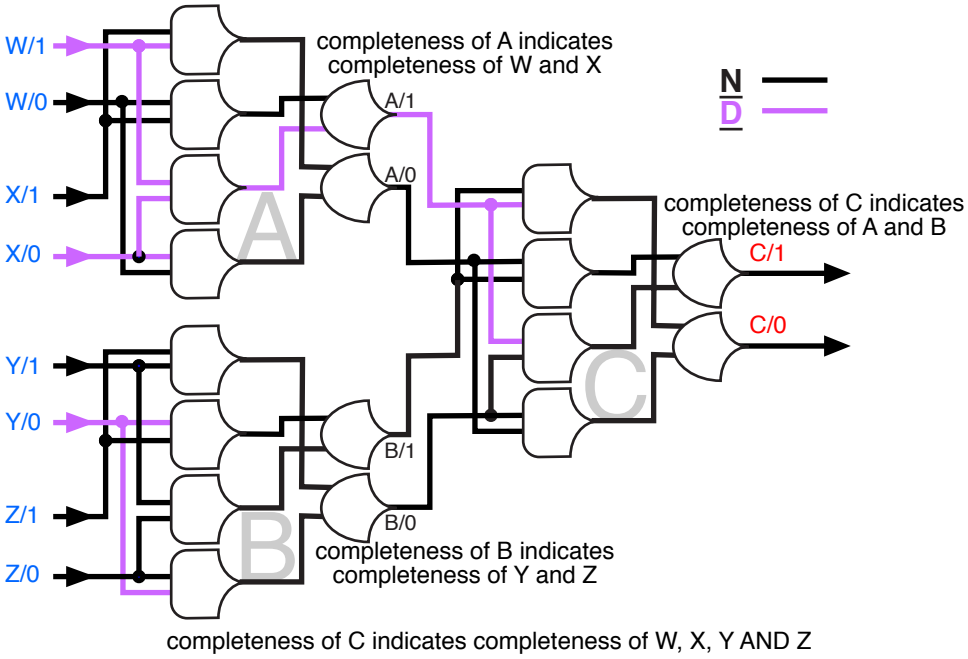


Figure 3.19. The composition of completeness relations in bignet.

3.4.10.5. Constancy

The completeness of **C** is dependent through **net** C on the completeness of **A** which is dependent through **net** A on the completeness of **W** and **X** and on the completeness of **B** through **net** B which is dependent on the completeness of **Y** and **Z**. If **A** is constant in relation to **W** and **X** and **B** is constant in relation to **Y** and **Z** and **C** is constant in relation to **A** and **B** then **C** is constant in relation to **W**, **X**, **Y** and **Z**.

3.4.10.6. The completeness criterion

In Figure 3.19 the completeness of **C** implies the completeness of its two inputs **A** and **B**. The completeness of **A** implies the completeness of **W** and **X**. The completeness of **B** implies the completeness of **Y** and **Z**. Then the completeness of **C** implies the completeness of **W**, **X**, **Y** and

Z. If any of W, X, Y and Z are not completely transitioned C will not be completely transitioned. This is the case for both the transition to **D** completeness and for the transition to completely **N**. Figure 3.19 illustrates a transition to D presentation that is not yet complete. W, X and Y have transitioned to D completeness but Z has not yet transitioned. C will transition to D completeness only after Z transitions to D completeness.

3.4.10.7. Concurrency

The transition to completeness of localities **A** and **B** are singularly appreciable events in relation to **net** A and **net** B respectively. However in relation to **bignet** they are independent and concurrent. The concurrent behaviors of **net** A and **net** B are coordinated by the completeness behavior of **net** C.

3.4.10.8. Instances of interaction, wholeness and chaos

As constant networks are composed into larger composite constant networks the newly component networks become dependent on the wavefront of the composite network for their input presentation. While each component network retains the same singularity of behavior in its own local context that it exhibited as a stand alone network each component singularity of behavior is now one singularity among many in the composite network as a whole. As an interaction wavefront flows through a composite network each component network and each component primitive behavior appreciates its own output transition to completeness marking its own local instance of interaction. Even though these instances of component interaction are all subordinate to and dependent on the flow of the composite network wavefront and are determined by dependency relations that converge to transition to completeness of the output locality of the composite network there is no singularly coherent relation among the collective component transition behaviors. The flowing wavefront that produces a singular transition event for the network as a whole (section 3.4.7.3) produces an incoherent chaos of local component network events in relation to the composite network as a whole.

As a constant network grows larger the extent of this internal incoherence increases within the network while the transition to completeness of composite network's output locality remains the only singularly appreciable behavior in relation to the network as a whole marking one instance of interaction (sections 3.4.6.5 and 3.4.7.4).

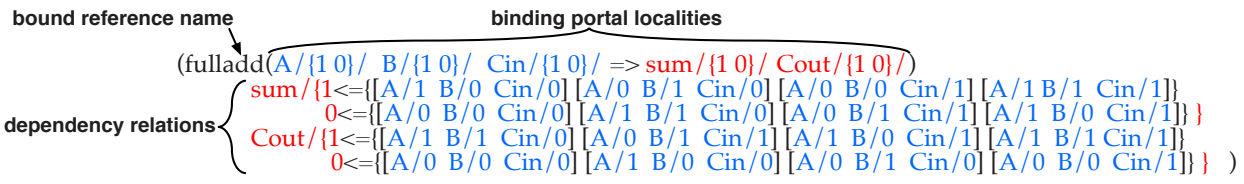
3.4.11. 5 bit adder constant network

The 5bit addition example illustrates a more complex constant network, that fulfills the completeness criterion and which can be indiscriminately referenced and copied.

3.4.11.1. component

The first step is to express the full adder and compose a 5 bit number locality then to compose the 5 bit adder in terms of the full adder and 5 bit number localities.

The fulladd expression specifies the fulladd dependency network in the left of Figure 3.20.



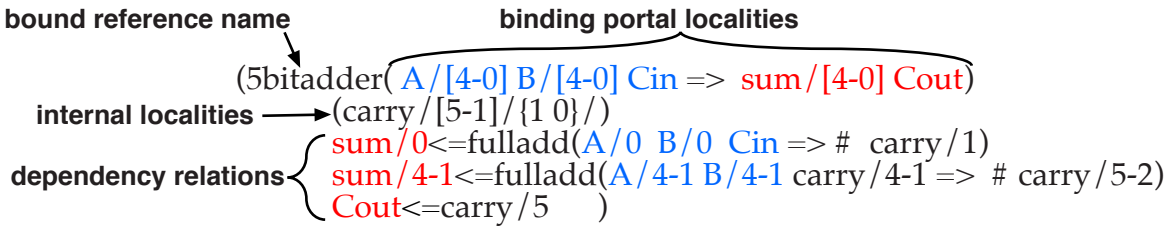
sum is dependent on the dependency of its components, **[1 0]** on input localities **A**, **B** and **Cin** distributed by the binding portal. **Cout** is dependent on the dependency of its components, **[1 0]** on input localities **A**, **B** and **Cin** distributed by the binding portal.

3.4.11.2. Composition

With **fulladd** an expression is encountered with two output assertions. This poses a difficulty with representing dependency with reference nesting as the reference nesting relation is one to one. Two versions of **fulladd** can be expressed, one with a **sum** output and one with a **Cout** output each of which can then be nested. Another approach with full portal reference is to designate the dependency of one output as nested with # in its syntactic position and to designate the dependency of the other output by name correspondence with a name in its syntactic position.

In the references to **fulladd** in the **5bitadder** expression the dependency relation of the assertion locality **sum** is designated to be nested, #, and the dependency relation of the assertion locality **Cout** is represented with name correspondence, **carry/x**.

The **5bitadder** expressed in terms of **fulladd** and the structure of a 5 bit number locality.

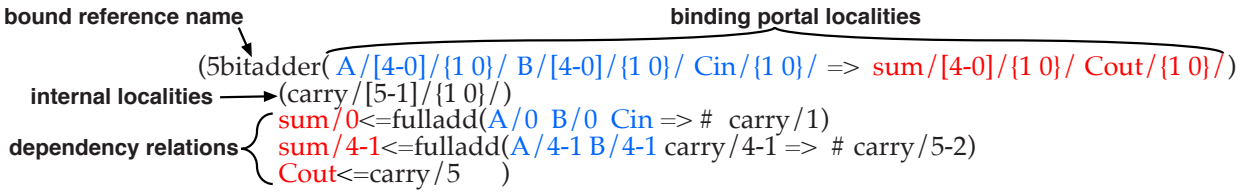


3.4.11.3. Inheritance

The locality expressions **A/[4-0]**, **B/[4-0]**, **sum[4-0]** and **Cout**, because they do not include the terminal dangling / or **/{D N},** are not complete. **A/[4-0]** just specifies **"all of"** five things named **4**, **3**, **2**, **1**, and **0**. What the things are will be inherited by reference.

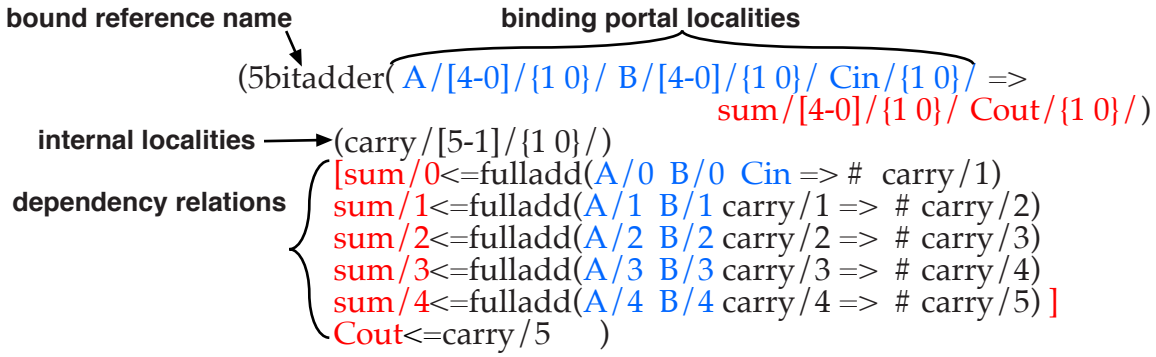
Each referential **A/4-1** expands to **A/4 A/3 A/2 A/1** with each **A/x** referencing an instance of **fulladd**. Each instance of **A/x** corresponds to **fulladd/A** and inherits from **fulladd/A** completing the referencing locality. The locality expression **A[4-0]** becomes **A[4-0]/{1 0}/** completely representing a 5 bit binary number. The inheritance could have been **A[4-0]/{2 1 0}/** representing a five trit number or **A[4-0]/{3-0}/** representing a five quat number.

The inheritance expansion from **fulladd**



sum/[4-0] expands to sum/4, sum/3, sum/2, sum/1 each referencing an instance of fulladd.

The above expression expands to:



3.4.11.4. Adder Network

The above expression maps to the **5bitadder** network in the right of Figure 3.20 composed in terms of **fulladd** component networks.

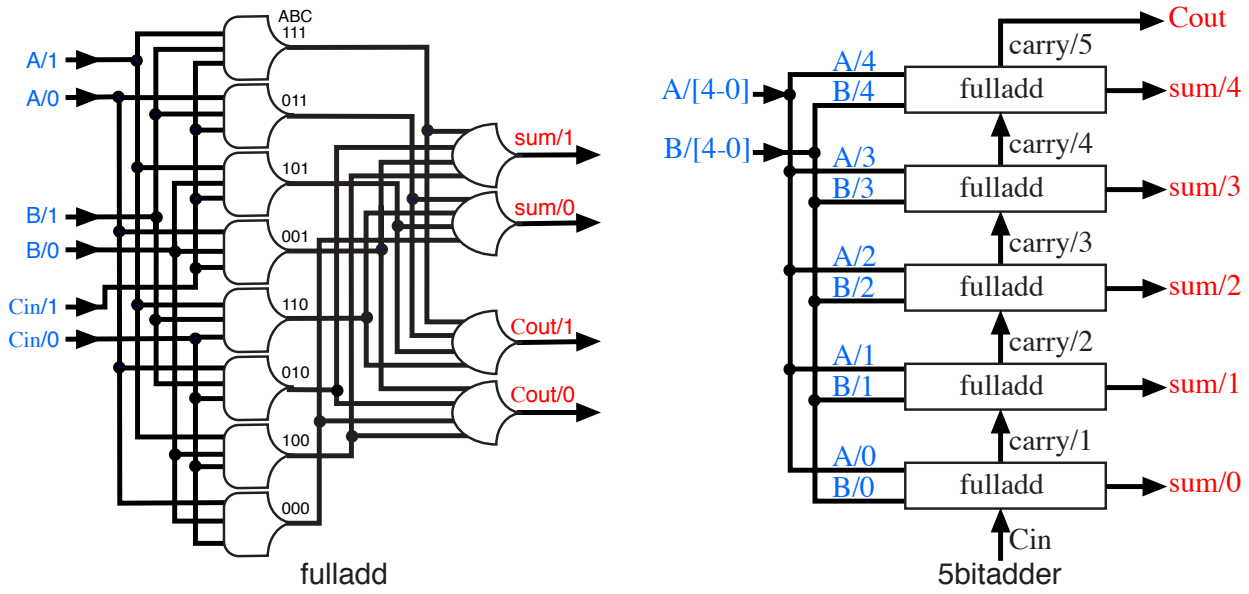


Figure 3.20. Fulladd network and 5 bit adder network.

3.4.11.5. Constancy and completeness

Each **fulladd** is constant and fullfils the completeness criterion. So **5bitadder** is constant and fulfillls the completeness criterion. Each successive **fulladd** in the **carry** chain of **5bitadder** requires the predecessor **carry** to meet completeness of input to transition its output. If any part of **A** or **B** is not complete then at least one **fulladd** will not achieve input completeness, not transition its **sum** and at least one part of the **5bitadder sum** will not transition to completeness. If the **5bitadder sum** transitions to completeness it implies that **A** and **B** and **Cin** have transitioned to completeness fulfilling the completeness criterion for the network as a whole.

The expression of a 5 bit multiplier constant network is presented in Appendix F.

3.5. INTERLUDE: The constant network

A constant network is **a network of dependency relations** among primitive behaviors and localities of differentness with the primitive behaviors linearly and progressively associated output to input i.e. no circular or feedback associations (section 3.4.2). It is *constant* in that it always asserts the same completeness of output differentness for the same presented completeness of input differentness (section 3.4). It *fulfills the completeness criterion* in that the transition of its output to completeness implies that the presented input has transitioned to completeness, that the interaction is complete and that the output is the correct result for the presented input (section 3.4.8.2). It is *indiscriminately composeable* in that it, can be referenced from and copied to anywhere and anywhen, in particular to compose bigger constant networks (section 3.4.10).

A constant network manifests interaction as *wavefronts of differentness transition* flowing through the network coordinated in terms of *completeness relations* (section 3.4.7). A wavefront is initiated by the transition of input presented from an *external environment* to an *exposed binding portal* which bounds the network with input and output ports (section 3.4.6). The constant network is subordinate to and is controlled by the input presented from the external environment which provides its source of liveness (with presentation), determines its temporal instance of interaction (with the monotonic transitions between **D** completeness and completely **N**) and provides its source of variability of behavior (The constant network's behavior varies only with the differentness of its presented input).

The responsible environment

For a constant network to behave the environment is still **required to monotonically transition inputs to the network with an appropriate delay** between the transition to **D** completeness and the transition to completely **N** to allow each consequent transition wavefront to flow completely through the network. This responsibility **is now relegated to the input presentation of the exposed binding portal of the constant network as a whole**. If fulfilled for the constant network as a whole then the monotonic transitioning and delay requirements for the primitive interaction behaviors and component constant networks within the composite constant network are also fulfilled.

By the end of this chapter the network will be in complete control of itself in relation to a passively indifferent environment.

3.5.1. Constant network space

The constant network bounded by its exposed binding portal is the interaction space.

3.5.2. Constant network time

A constant network can perform only one interaction at a time. The transition of the its output locality to **D** completeness implying that the presented input is complete and that the output is

the correct result of the presented input is the only singularly appreciable network behavior bounded by the exposed binding portal and framed by transition to completely **N** wavefronts coherently marking one instance of interaction and one instance of interaction time for the network as a whole (sections 3.4.7.4 and 3.4.7.3).

3.5.2.1. Extending association differentness through differentness of time

Even if successive interactions are identical in all aspects of transition behavior the interactions are different by virtue of differentness of instances of interaction time .bounded by transition to completely **N**. Each instance of interaction, each reuse of a constant network, extends its expression of differentness of interaction through differentness of time. This is temporal differentiation (Chapter 4).

3.5.2.2. Ephemeral time

However, these instances of interaction are ephemeral (section 3.4.7.4), disappearing from the network, never associating and never interacting in the context of the network which cannot account this extension of expression of differentness of interaction.

3.5.2.3. Interaction incoherence

Constant network individual inputs can transition to completeness in varying orders including all at once. Wavefront transitions flowing through the network concurrency relations with varying delays can transition in varying orders including all at once (section 3.4.7.3 and 3.4.10.8). There is no singular referent of time or space that coherently relates them. Wavefront flow through the network is coordinated in terms of dependency completeness relations not in terms of temporal relations or spatial relations.

3.5.2.4. Wholeness

The exposed binding portal bounds the network providing its liveness and time and realizing the network's reuse determines the wholeness of the network, its instances of interaction and of interaction time (sections 3.4.6.5 and 3.4.10.8). This wholeness of constant network is not wholeness of expression which leaks through the binding portal into the external environment not accounted by the constant network.

3.5.3. Nothing new

A constant network is a linear progression of specifically interacting differentnesses represented as a directed network of dependency relations among the primitive behaviors defined in section 3.2. Nothing new has been introduced either abstractly or substantially.

3.5.4. The transcendental view

Because a constant network always asserts the same result differentness for the same presented input differentness, a constant network, however complex, can be abstractly characterized as a single step mapping transcending and ignoring its internal structure and behavior which includes intrinsic concurrent dependency relations. This equates the network with a single step, single place primitive behavior. This reduction of complexity to primitivity rather overlooks the emergence of complexity from primitivity.

3.6. QUANDARY 5: The environment

On the one hand the external environment can be incorporable into a constant network. On the other hand the external environment is never fully incorporated. What and where is this inscrutable external environment that is in complete control of the constant network, residing always on the other side of its exposed binding portal, undoubtably there, but always receding just beyond grasp like the never findable end of the rainbow?

This binding portal exposed to an uncharacterized environment is intrinsic to constant networks. If a constant network were not completely dependent on its presented input it would not be indiscriminately referenceable and copyable. But the quandary remains. *Key aspects of a constant network, its variability of behavior, its time and its very liveness lie beyond its expressivity.*

3.7. The oscillation network: self regulation

A constant network fulfilling the completeness criterion can use the completeness transitioning of its output to self regulate wavefront flow into the network. In Figure 3.21 the completeness of the output transition (in this case the transition of **"one of"** **C/0** or **C/1** to **D** or the transition of **C** to **"all of"** **N**) is appreciated by a **"one of"** behavior which reduces the completeness of output to a single place of association **C.comp** with condition **D** representing **D** completeness and condition **N** representing completely **N**. **C.comp** is the singular appreciation of the singularly appreciable completeness of the output locality. This singular appreciation of completeness of interaction corresponds to the **Z** condition of the Roman numeral example of Chapter 2 (section 2.4.4).

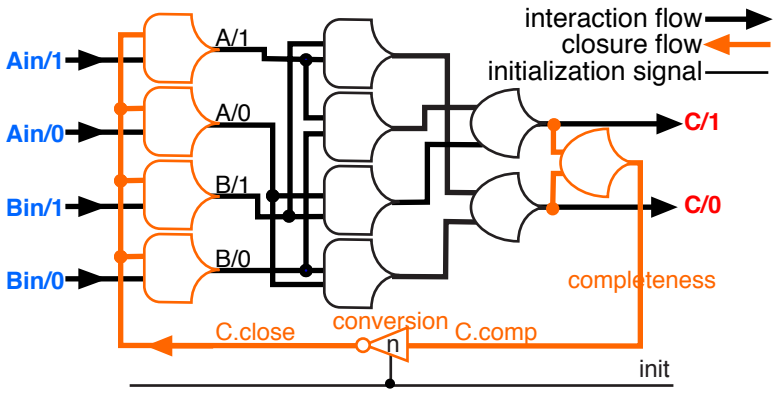


Figure 3.21. Self regulating oscillation network.

The dependency relations for appreciating and reducing the completeness of a locality to a single place of association are contained in the expression of each locality (section 3.9.2).

3.7.1. Regulating network input

The appreciation of output completeness **C.comp** is converted ~**C.comp**=>**C.close** and closes with the input through **"all of"** behaviors to allow the next transition wavefront into the network. With the network empty and its output completely **N** **C.comp** will be **N** and

~C.comp=>C.close will be **D** which will enable a wavefront of transition to **D** completeness to flow through the rank of **"all of"** behaviors into the constant network. C.close will remain **D** with the **"all of"** behaviors maintaining the **D** completeness wavefront until the wavefront has propagated all the way through the network and transitioned the output to **D** completeness. If Ain and Bin transition to completely **N** they will wait for C.comp to transition to **N**.

When the output transitions to **D** completeness C.close transitions to **D** and ~C.comp=>C.close transitions to **N** allowing a transition to completely **N** wavefront into the constant network. If the input is not completely **N** the rank of **"all of"** behaviors will wait for the input to transition to completely **N**. C.close will remain **N** with the **"all of"** behaviors maintaining the **N** wavefront until the wavefront has propagated all the way through the network and transitioned the output to completely **N**. If Ain and Bin transition to **D** completeness they will wait for C.close to transition to **D**.

When the output transitions to **N** completeness C.comp transitions to **N** and ~C.comp=>C.close transitions to **D** allowing a transition to **D** completeness wavefront into the constant network. If the input is not **D** completeness the rank of **"all of"** behaviors will wait for the input to transition to **D** completeness. When the transition to **D** completeness wavefront arrives it passes through the **"all of"** behaviors into the constant network. C.close will remain **D** with the **"all of"** behaviors maintaining the **D** completeness wavefront until the wavefront has propagated all the way through the network and transitioned the output to **D** completeness. If Ain and Bin transition to completely **N** they will wait for C.close to transition to **N**.

When the output transitions to **N** completeness C.comp transitions to **N** and ~C.comp=>C.close transitions to **D** allowing a transition to **D** completeness wavefront into the constant network.

And so on.

The next input transition is not allowed into the network until the current transition wavefront has completely propagated through the network and transitioned the output. The closure, recognizing the singularly appreciable event of the output of the constant network transitioning to completeness manages the monotonic transitioning of input presentations regulating the flow of wavefronts into the network.

3.7.2. Expressing the oscillation network

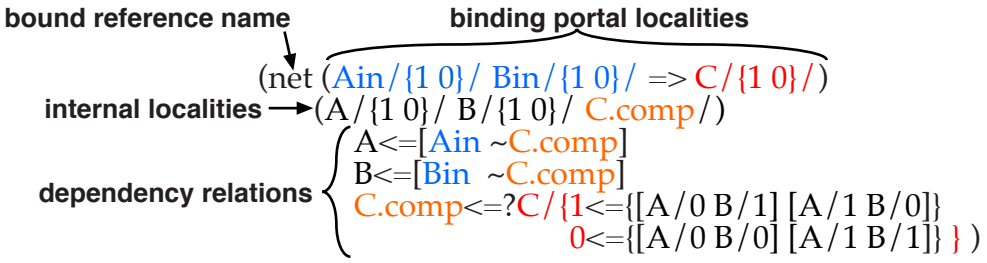
The closed network with a single converted differentness appreciating the monotonic transitions or oscillations between two disjoint completeness representations is similar to a spontaneously alive binary ring oscillator circuit with a single inversion. So the network is called an *oscillation network*.

The **black** paths in Figure 3.21 are interaction wavefront flow paths. The **orange** paths are closure flow paths. The initializing converter allows the network to be initialized to all **N**, empty of interaction differentness ready for the first transition to **D** completeness (appendix section D.1.1.)

The network expression with closure introduces two syntax elements. The tilde ~ represents conversion of **D** to **N** or of **N** to **D** (Section 3.2.4). The conversion occurs only in closure, never in an interaction wavefront. The **?** before a locality name indicates the completeness reduction of

the referenced locality to a single place of association of the same name with a **.comp** suffix, **name.comp**.

Expression of the oscillation network



C.comp is dependent on the completeness of **C** the completeness of which is dependent on the components of **A** which are dependent on **Ain** and conversion of **C.comp** and on the components of **B** which are dependent on **Bin** and conversion of **C.comp.**

3.8. INTERLUDE: Marking time with the oscillation network

Closing a constant network with the converted output locality completeness enabling the network input creates a spontaneously alive, self regulating network continually striving to transition between **D** completeness and completely **N** performing the interaction of its encompassed network on successive presentations of completeness transition with its own appropriate delay.

The encompassed interaction network provides *interaction behavior*. The input **"all of"** enable rank provides *memory behavior*. The completeness criterion and the completeness closure provide *coordination behavior*. These are the same properties exhibited by the primitive behaviors in section 3.3. At this point the oscillation network can be considered an abstract primitivity. All networks from now on will be composed of oscillation networks.

The encompassed interaction network must be constant. If a wavefront in the network flows anywhere other than to contribute to the output of the network the completeness of the output of the network will not imply the status of the non contributing wavefront.

The responsible environment
At this point the closure is intrinsic to the expression. The external environment does not have access to the closure so must still time itself in relation to its own time metric and with understanding of the network delays. The environment transitions cannot get ahead of the intrinsic closure enables. The environment does not have access to the intrinsic closure behavior and there is no explicit coordination between the oscillation expression and the environment. It may seem that the emergent oscillation network has not accomplished anything useful, but stay tuned.

3.8.1. Oscillation network space and time

The constant network encompassed by the closure is the interaction space. The transitions of the output completeness between **D** completeness and completely **N** are the appreciable boundaries of instances of interaction and differentness of interaction time in relation to the

encompassed constant network. The oscillation network appreciating these boundaries serves as an *escapement mechanism* establishing a nonuniform metric of interaction time for its encompassed constant network in relation to its flowing wavefronts of transition. The oscillation network is now in control of its own interaction time and there is no longer any appeal to a conventional metric of time. Network interaction time becomes a metric of time in itself unrelated to any conventional metric of time.

From this point on networks will be in control of their own time.

3.8.2. Nothing new

An oscillation network is an emergent behavior expressed solely in terms of dependently associating the defined primitive behaviors (section 3.2). Nothing new has been introduced either abstractly or substantially.

3.8.3. No external metrics

There is no coherent reference frame or external metric of space or of time relative to an oscillation network and trying to impose an external metric onto the oscillation network contributes nothing to either the understanding of or the realization of the oscillation network.

3.8.4. The transcendental view

The closure network can be removed and ignored and the constant network within the closure network viewed as a single step mapping behavior if one wishes.

3.9. The pipeline network: composing self regulation

A pipeline network is a structure of linked oscillation networks coordinating their wavefront flow with closure relations. Placing the output completeness determination of oscillation network **A** after the input enable closure of oscillation network **B** links the two oscillation networks coordinating wavefront flow from oscillation network **A** to oscillation network **B**.

Figure 3.22 shows a **pink** an **orange** and a **blue** oscillation network. They are linked by placing the completeness of the **pink** network after the enable of the **orange** network and by placing the completeness of the **orange** network after the enable of the **blue** network forming a pipeline of linked oscillation networks.

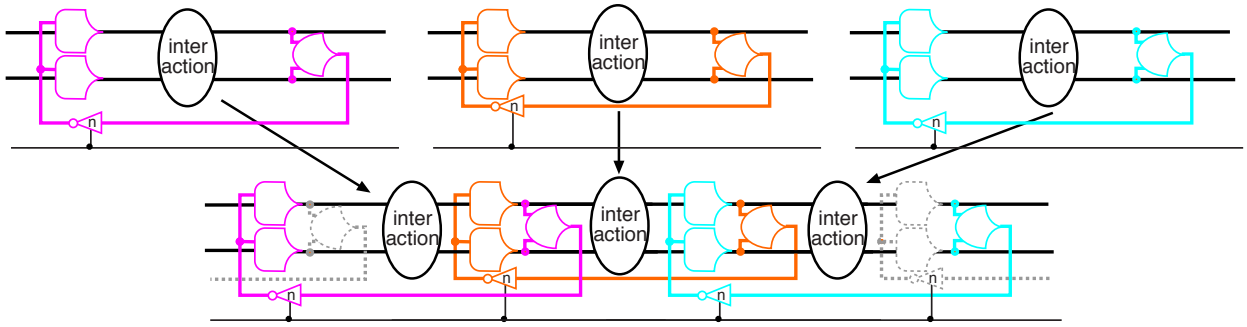


Figure 3.22. linking oscillation networks.

The wavefront accepted by the **pink** network from a previous network flows to the enable of the **pink** network and is stably maintained until accepted by the **orange** network however long

this might take. When accepted by the **orange** network the **pink** network can determine the completeness of its flow and can accept the next wavefront transition into its network.

The wavefront accepted by the **orange** network from the **pink** network flows to the enable of the **blue** network and is stably maintained by the **orange** network until accepted by the **blue** network however long this might take. When accepted by the **blue** network the **orange** network can determine the completeness of its flow and can accept the next wavefront transition into its network.

The wavefront accepted by the **blue** network from the **orange** network flows to the enable of a next network and is stably maintained by the **blue** network until accepted by the next network however long this might take. When accepted by the next network the **blue** network can determine the completeness of its flow and can accept the next wavefront transition from the **orange** network into its network.

And so on.

3.9.1. Pipeline wavefront flow

Each oscillation network of a pipeline network is individually and spontaneously alive continually striving to transition between **D** completeness and completely **N**. The oscillation networks mutually coordinate each other's striving through the closure links. At each closure link an early interaction wavefront transition will wait indefinitely for a corresponding closure enable transition and an early closure enable transition will wait indefinitely for a corresponding interaction wavefront transition. As oscillation networks individually oscillate, alternating wavefronts of transition to **D** completeness and completely **N** spontaneously flow, fully coordinated, from oscillation network to oscillation network through the pipeline network with transition to **D** wavefronts interacting as they flow through the constant networks encompassed within the oscillation networks.

3.9.2. The closure link

Placing the completeness for a locality of a delivering oscillation network after the enable of the same locality of the receiving oscillation network forms a closure **link**. A closure link for a locality is determined by the expression of the locality which specifies the determination of its transition to **D** completeness and its enable completeness.

The closure link enable network is a rank of "all of" behaviors The closure link completeness network is determined by the locality expression.

```
B/{1 0}/          /*  drop the terminal slash */
B/{1 0}           /*  then expands to */
{B/1 B/0}
```

Expressing the completeness network for the one bit link of Figure 3.23


```
B/[1 0]/{1 0}/          /*  drop the terminal slash */
B/[1 0]/{1 0}           /*  then expands to */
B/[1/{1 0} 0/{1 0}]             /* then expands to */
B/[{1/1 1/0} {0/1 0/0}]                 /* then expands to */
[{B/1/1 B/1/0} {B/0/1 B/0/0}]
```

Expressing the completeness network for the two bit link of Figure 3.23

```
B/[4-0]/{1 0}/          /*  drop the terminal slash */
B/[4-0]/{1 0}           /*  then expands to */
B/[4/{1 0} 3/{1 0} 2/{1 0} 1/{1 0} 0/{1 0}]             /* then expands to */
B/[{4/1 4/0} {3/1 3/0} {2/1 2/0} {1/1 1/0} {0/1 0/0}]           /* then expands to */
[{B/4/1 B/4/0} {B/3/1 B/3/0} {B/2/1 B/2/0} {B/1/1 B/1/0} {B/0/1 B/0/0}]
```

Expressing the completeness network for the five bit link of Figure 3.23

```
B/{3 2 1 0}/            /*  drop the terminal slash */
B/{3 2 1 0}             /*  expands to */
{B/3  B/2  B/1  B/0}
```

Expressing the completeness network for the quaternary link of Figure 3.23

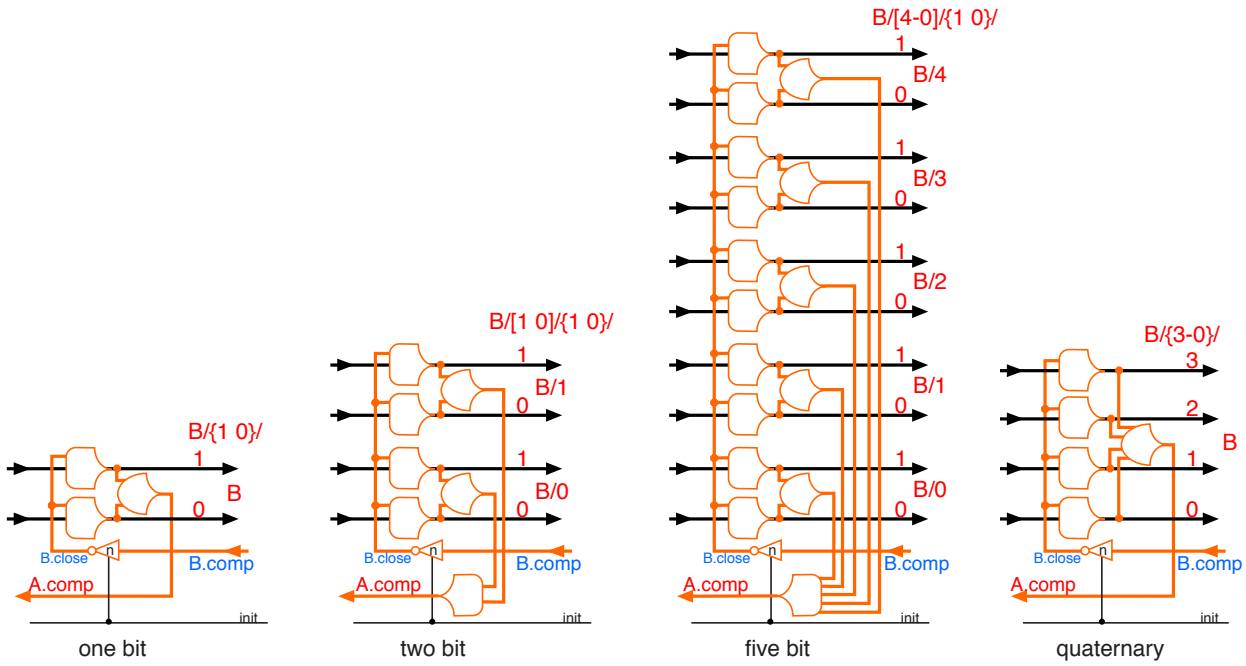


Figure 3.23. Completeness of localities with their structure expressions.

See the localities of **net** Figure 3.25 for one bit locality links. See **qtob/B** Figure 3.27 for a two bit locality link. See **5bitadd** section 3.4.11 for five bit locality links. See **qtob/A** Figure 3.27 for a quaternary locality link.

The closure link is similar to the notion of asynchronous handshakes.

> "in its most general form, asynchronous design removes the global clock in favor of distributed local handshaking to control data transfer and changes of state."

This is a view that defers to clocked Boolean design as the fundamental referent with asynchronous design as a subordinate derivative variation. The present narrative takes the opposite view considering dependently flowing asynchronous behavior as fundamental and synchronous behavior as derivative (sections 1.1.1, 3.4.7 and 5.7.5). Closure represents a more integrated wholeness than does the notion of handshake.

3.9.3. The closure protocol

- A locality completeness closes with (enables monotonic transition of) all localities on which its completeness is dependent.
- A locality is closed by (its monotonic transition enabled) all the localities which depend on it for their own completeness transition.

A network typically has two or more input localities and one output locality which is dependent on all of the input localities. So the completeness of the output closes with all the inputs as shown in Figure 3.21 forming an oscillation network. The inputs, however, are two different localities from two different networks and must be viewed as closed individually. The oscillation network on the left of Figure 3.24 is a redraw of Figure 3.21 to show the localities **A** and **B** closed individually.

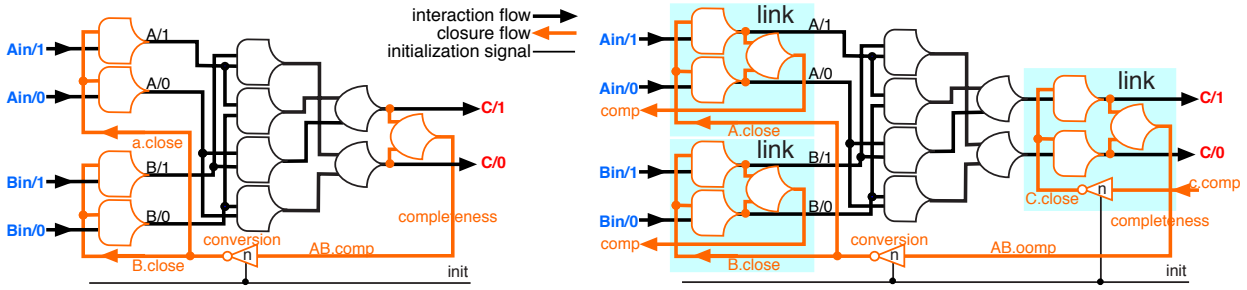


Figure 3.24. Localities A and B are shown as individually closed localities.

Each locality **A** and **B** will determine its own completeness and close with the localities on which it is dependent as shown on the right of Figure 3.24. Locality **C** contributes to the completeness of some locality and will be closed with and enabled by that locality as shown on the right of Figure 3.24. Links appear on the input and output of the oscillation network.

There is a problem with expressing the links in a each link is expressed by different network expressions requiring an interrelationship between network expressions expressions complicating the expression of their composition. This is addressed by having each network express a link only for its output locality which will be referred to as a half oscillation.

Figure 3.25 illustrates the network at the right of Figure 3.24 with only its output link expressed. The links for its two inputs are now output links in separate networks. Closure relations now flow beyond the oscillation network expression and necessarily pass through its binding portal closing with output links in other networks.

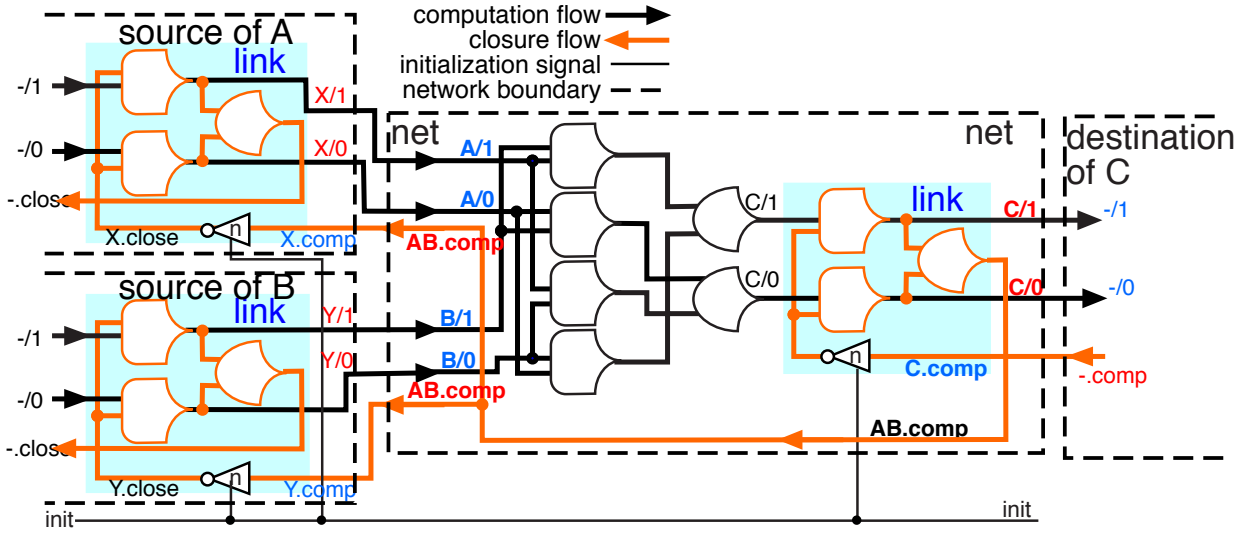


Figure 3.25. Constant network encompassed with explicit closure links.

3.9.4. Counter flowing networks

Figure 3.25 and Figure 3.22 provide the first glimpses of the counter flowing networks of a pipeline network shown in Figure 3.26. A pipeline network composed of oscillation networks forms a structure of interpenetrating counter-flowing networks mutually regulating each other. Interaction transition wavefronts flow through the **black** interaction network in the direction of interaction. Closure transition bubbles flow through the **orange** closure network counter to the direction of interaction flow. The counter flowing networks intersect through localities with links that coordinate the counter directional flows with common singularly appreciable completeness relations (section 3.9.7). The wavefronts spontaneously flow into bubbles as the bubbles flow in the reverse direction around the wavefronts (Appendix D).

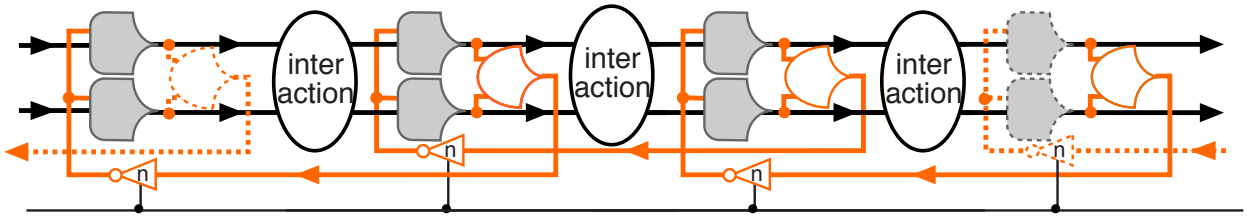
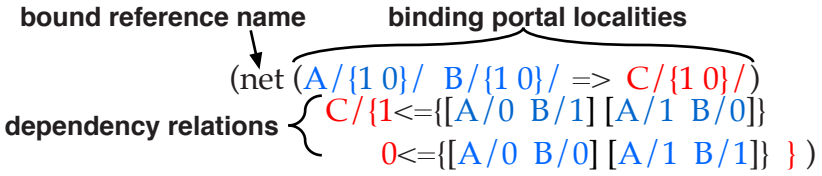


Figure 3.26. Counter flowing networks.

3.9.5. Expressing closure flow and composition

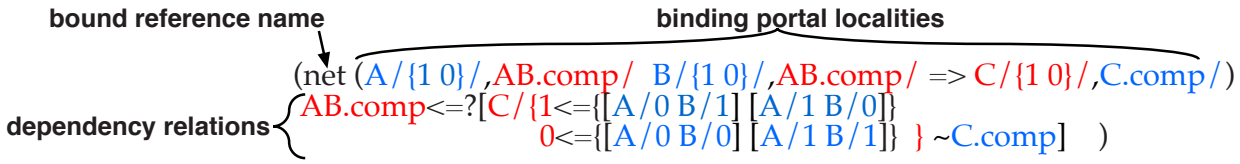
The expression of the network named **net** in Figure 3.25 begins with the expression of the enclosed network from Figure 3.12.

The expression of the constant network encompassed by the oscillation network.



A half oscillation expresses the interaction dependencies and the enable and completeness of its result locality link as well as the flows through its binding portal. Each locality in the binding portal is now associated with a closure locality by a comma. **A/{1 0}/,AB.comp/** expresses the interaction locality **A/{1 0}/** associated with its completeness closure **AB.comp/**. The colors still represent the direction of transition flow through the binding portal, blue is input and red is output. **C/{1 0}/,C.comp/** expresses the result locality **C/{1 0}/** associated with its completeness closure **C.comp/**. Again **blue** is input and **red** is output.

The interaction dependency relations are inside an **"all of"** behavior with **~C.comp** which is the last dependency applied before the completeness of result locality **C** is appreciated.



AB.close is dependent on the completeness reduction **?** of **C** the completeness of which is dependent on the completenesses of **A** and **B** and the conversion **~** of **C.comp.**

The interaction and closure flows through a portal are dependent but are not simultaneous. For instance, in the binding portal **A** is paired with **AB.comp**. In Figure 3.25 an **AB.comp** transition will flow out of the portal and later enable the transition of **X** which will then flow into the portal as a transition of **A**.

3.9.5.1. Half oscillations

There is one complete oscillation network in Figure 3.25 but the expression of the oscillation network occurs over three component half oscillation expressions none of which contains a complete oscillation network. Each component constant network with a *link* on its output locality forms a *pipeline component network* expressing two *half oscillations*, an input/completeness half oscillation closing with the inputs to the network and an output/enable half oscillation closed by the output of the network.

The responsible environment

The environment now has access to the closure through the binding portal and the responsibility of the environment becomes to hold each interaction locality transition until the completeness of transition of the result locality is indicated by its closure. The environment no longer need understand the delay behavior of the expression in terms of its own time metric but now the **oscillation network itself determines the appropriate delay between transitions of input presentation.** The responsibility of the environment is now to honor the closure by next transitioning the input only after each transition wavefront is accepted with closure by the network.

The environment is still a source of variability and monotonic transitioning but it is no longer in complete control. The network itself is now in control of its own time. There is no longer any appeal to a conventional metric of time.

3.9.6. Composing half oscillations

Pipeline composition is illustrated by adding links to the output locality of the constant networks **qtob** and **btoq** of section 3.4.10 forming the pipeline component networks of Figure 3.27.

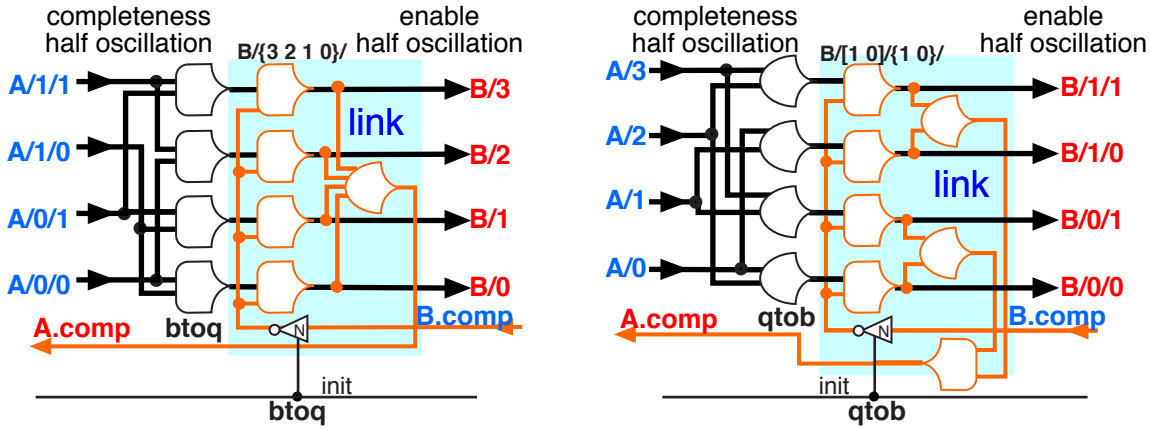


Figure 3.27. Primitive pipeline component networks bounded by half oscillations.

Expression for the half oscillation pipeline component **btoq**.

bound reference name **binding portal localities**

```
(btoq(A/[1 0]/{1 0}/,A.comp/ => B/{3 2 1 0}/,B.comp/)
```

dependency relations

```
A.comp<=?[B/{3<=[A/1/1 A/0/1]
          2<=[A/1/1 A/0/0]
          1<=[A/1/0 A/0/1]
          0<=[A/1/0  A/0/0]} ~B.comp] )
```

A.comp is dependent on the completeness reduction **?** of **B** the completeness of which is determined by the dependency relations of its components, **{3 2 1 0}** on input locality **A**

components and ~B.comp. In the binding portal input/presentation locality A is paired with closure A.comp and output/assertion locality B is paired with closure B.comp.

Expression for the half oscillation pipeline component **qtob**.

bound reference name — binding portal localities

```
(qtob(A/{3 2 1 0}/,A.comp => B/[1 0]/{1 0}/,B.comp)
    A.comp<=?[B/[1/{1<={A/3  A/2}
                   0<={A/1  A/0}}
              0/{1<={A/3  A/1}
                 0<={A/2  A /0}}]  ~B.comp ] )
```

dependency relations

A.comp is dependent on the completeness reduction **?** of **B** the completeness of which is determined by the dependency relations of its components, **[1 0]/{1 0}** on input localities **A** components and ~ **B.comp**. In the binding portal input/presentation locality **A** is paired with closure **A.comp** and output/assertion locality **B** is paired with closure **B.comp**.

3.9.6.1. Associating the half oscillations

The example pipeline network is composed by associating the output/enable half oscillation locality **btoq/B** to the input/completeness half oscillation locality **qtob/A** forming a pipeline network containing one oscillation network and bounded by half oscillations. The pipeline network is composed as usual by associating a binding portal output to a binding portal input (Figure 3.18) but now the composition involves associating the paired closures which simultaneously composes the counter flowing networks

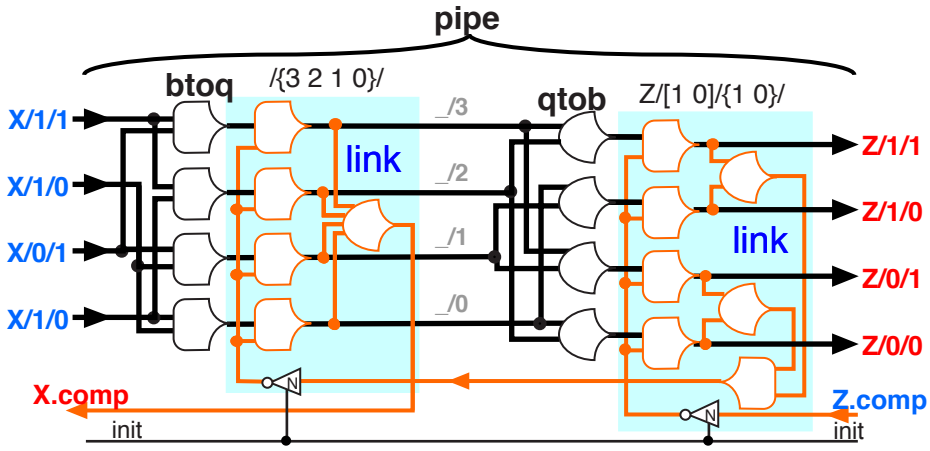
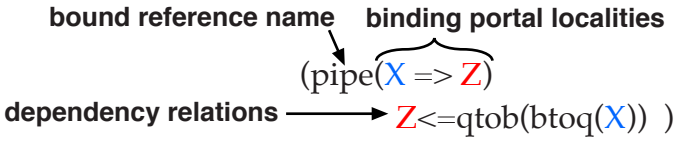


Figure 3.28. linked pipeline segment networks forming a pipeline network.

3.9.6.2. Locality inheritance

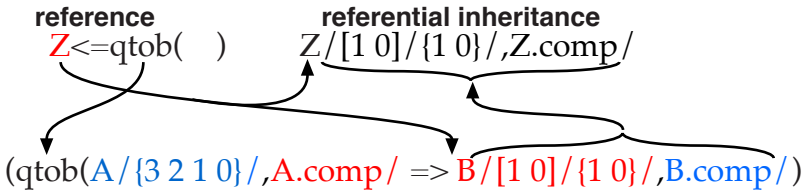
The expression of composition, Figure 3.28, is simplified by expressing the interaction flow dependency relations in terms of whole locality references and nameless nesting relations.

The dependency expression for **pipe**.

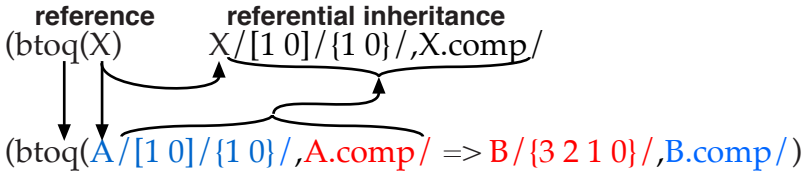


and inheriting closure relations as well as locality structure from the referenced component network expressions (section 3.4.10.3).

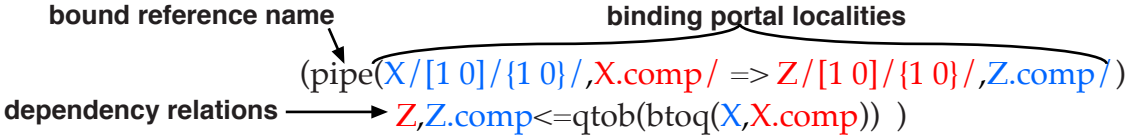
Z inherits locality structure and closure relation from **qtob/B**.



X inherits locality structure and closure relation from **btoq/A**.



The inheritance expanded expression for **pipe** network of Figure 3.28.

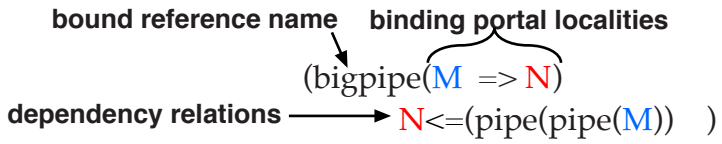


pipe/Z,Z.comp is dependent on **qtob/B,B.close** which is dependent through **btoq** on **pipe/X,X.comp.**

3.9.6.3. Composing bigger pipelines

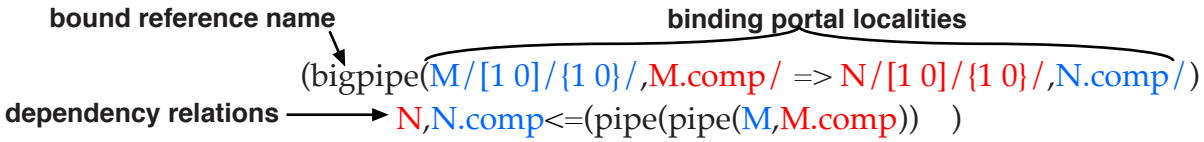
Pipeline networks bounded by half oscillations are composed by associating half oscillation binding portal localities to form larger pipeline networks. The expression below connects two **pipe** networks to form **bigpipe** network of Figure 3.29.

The dependency expression for **bigpipe** network of Figure 3.29



N is dependent on the outer **pipe** which is dependent on the inner **pipe** which is dependent on **M.** N inherits its locality structure and closure flow from **pipe/Z** which inherited its structure and closure flow from **qtob/B.** M inherits its locality structure and closure flow from **pipe/X** which inherited its structure and closure flow from **btoq/A**.

The inheritance expanded expression for **bigpipe** network of Figure 3.29.



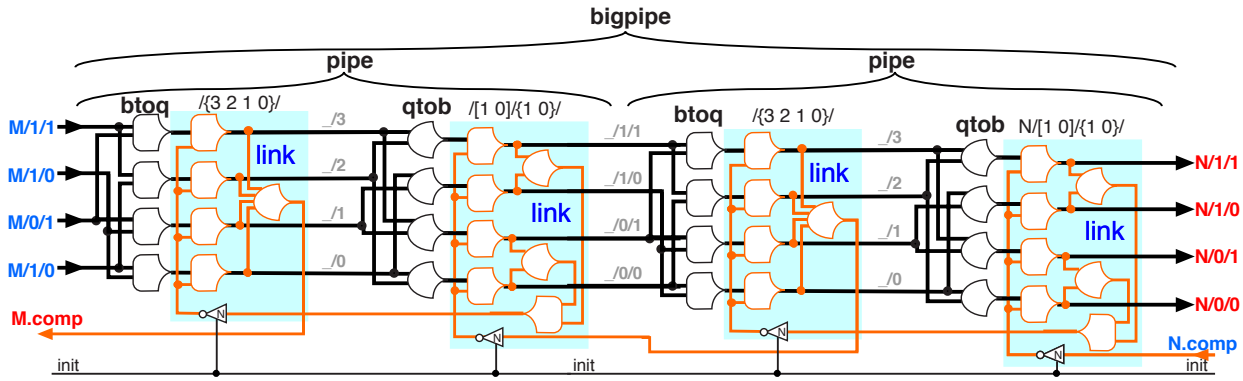


Figure 3.29. Longer pipeline composed of shorter pipelines.

The pipeline **bigpipe** contains three complete oscillation networks and is bounded by half oscillations ready for further composition.

3.9.6.4. Simultaneously flowing instances of interaction

Each transition to **D** completeness presented to an input half oscillation boundary of a pipeline network initiates a wavefront of transition to **D** completeness flowing through the pipeline network. After the **D** wavefront propagates through one or more oscillation networks the input completeness half oscillation boundary can accept a presentation transition to completely **N**. After the transition to completely **N** wavefront propagates through one or more oscillation networks the input completeness half oscillation boundary can accept a next presentation of transition to **D** completeness initiating a next transition to **D** completeness wavefront into the pipeline network. A pipeline network maintains the independence and integrity of multiple transition to **D** completeness wavefronts separated by transition to completely **N** wavefronts

simultaneously flowing through different stages of a long pipeline network each representing an isolated and discrete differentness of instance of interaction behavior and instance of interaction time.

The wavefronts remain isolated because a completely **N** wavefront can only flow into a **D** bubble. It cannot flow into a **D** completeness wavefront. Similarly a **D** completeness wavefront can only flow into a **N** bubble. It cannot flow into a completely **N** wavefront. The wavefronts can never overtake each other in a pipeline network (Appendix D).

3.9.6.5. Pipeline network time

Each successive transition to **D** completeness wavefront in a pipeline network is a future in relation to its predecessor transition to **D** completeness wavefronts and a past in relation to its successor transition to **D** completeness wavefronts.

3.9.6.6. The completeness criterion

The first transition to **D** completeness wavefront to flow into a pipeline network produces the first transition to **D** completeness wavefront flowing out of the pipeline network the completeness of which which implies the completeness of the first input presentation. The second input transition to **D** completeness wavefront produces the second output transition to **D** completeness wavefront which implies the completeness of the second input presentation and so on. If each half oscillation network is constant and fulfills the completeness criterion then the pipeline network as a whole is constant and fulfills the completeness criterion.

3.9.7. Variations of closure structure

The examples in Figure 3.30 illustrates the possible web granularities of counter flowing closure structures for the constant network of Figure 3.19.

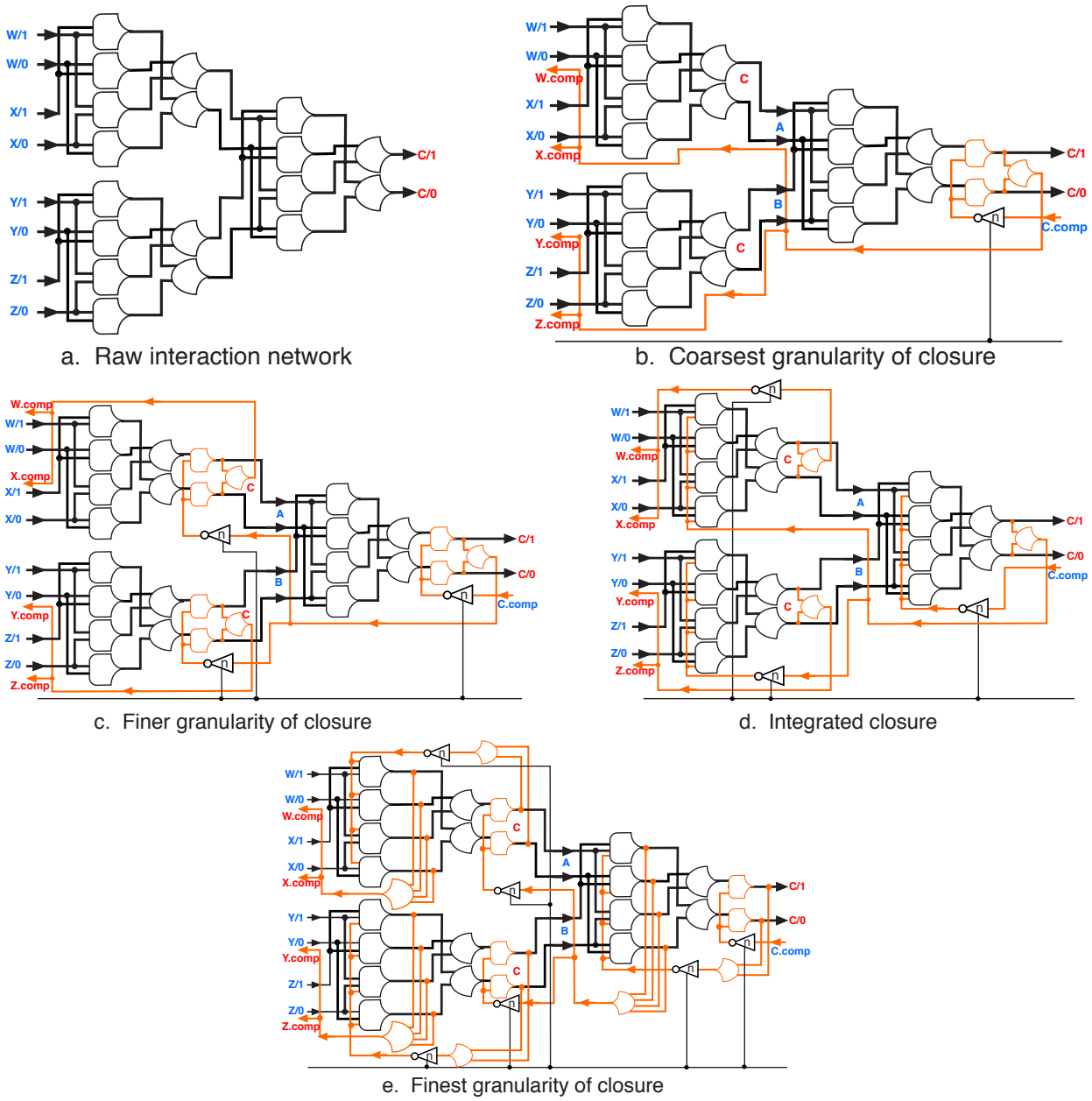


Figure 3.30. The varieties of closure structure for a given constant network.

3.9.7.1. Figure 3.30a network expression: raw constant network

The raw constant network as described in section 3.4.10.4 can be expressed as a composition of component networks:

```
(net(A/{1 0}/  B/{1 0}/ => C/{1 0}/)
        C/{1<={{[A/0  B/1] [A/1  B/0]}
           0<={[A/0  B/0] [A/1  B/1]} }  )
```


The **bignet** network of Figure 3.30a is expressed as references to **net**.
The dependency expression

```
(bigneta(W X Y Z => C)
    C<=net(net(W X) net(Y Z))   )
```

The inheritance expanded network expression.

```
(bigneta(W/{1 0}/ X/{1 0}/ Y/{1 0}/ Z/{1 0}/ => C/{1 0}/)

    C<=net(net(W X) net(Y Z))   )
```

Or the network can be expressed entirely in terms of relations among primitive behaviors with no references and no inheritance. Locality structures have to be explicitly expressed.

```
 (bigneta(W/{1 0}/  X/{1 0}/  Y/{1 0}/  Z/{1 0}/  => C/{1 0}/)
      C/{1<={ [ {[W/0  X/1] [W/1  X/0]}  {[Y/0  Z/0] [Y/1  Z/1]} ]
              [ {[W/0  X/0] [W/1  X/1]}  {[Y/0  Z/1] [Y/1  Z/0]} ] }
         0<={ [ {[W/0  X/1] [W/1  X/0]}  {[Y/0  Z/1] [Y/1  Z/0]} ]
              [ {[W/0  X/0] [W/1  X/1]}  {[Y/0  Z/0] [Y/1  Z/1]} ] } }  )
```

bigneta is a constant network with no closure.

3.9.7.2. Figure 3.30b network expression: coarsest granularity of closure

There are two approaches to expressing **bignetb** of Figure 3.30b.

1. The link can be syntactically incorporated into the primitive behavior expression of **bigneta**. There are no references to already expressed networks and hence no inheritance so **bignetb** has to explicitly express the closure in detail including the structure of the binding portal.

```
(bignetb(W/{1 0}/,WXYZ.comp/  X/{1 0}/,WXYZ.comp/  Y/{1 0}/,WXYZ.comp/  Z/{1
0}/,WXYZ.comp/ => C/{1 0}/,C.comp/)
WXYZ.comp<=?[C/{1<={ [ {[W/0  X/1] [W/1  X/0]}  {[Y/0  Z/0] [Y/1  Z/1]} ]
                     [ {[W/0  X/0] [W/1  X/1]}  {[Y/0  Z/1] [Y/1  Z/0]} ] }
                0<={ [ {[W/0  X/1] [W/1  X/0]}  {[Y/0  Z/1] [Y/1  Z/0]} ]
                     [ {[W/0  X/0] [W/1  X/1]}  {[Y/0  Z/0] [Y/1  Z/1]} ] } }  ~C.comp ] )
```

2. The link can be integrated into the expression of a component network and the component network incorporated by reference. Network **netL** incorporates a closure link to the above network **net**.

```
(netL(A/{1 0}/,AB.comp/  B/{1 0}/,AB.comp/ => C/{1 0}/,C.comp/)
    AB.comp,<=?[C/{1<={{[A/0  B/1] [A/1  B/0]}
                      0<={[A/0  B/0] [A/1  B/1]} }   ~C.comp] )
```

bignetb references **netL** with a link once and **net** without a link twice.
The dependency expression.

(bignetb(W X Y Z => C)
C<=netL(net(W X) net(Y Z)))

The entire dependency structure including the references to **net** and the binding portal inherits from **netL**.

The inheritance expanded network expression.

(bignetb(W/{1 0}/,W.comp/ X/{1 0}/,X.comp/
Y/{1 0}/,Y.comp/ Z/{1 0}/,Z.comp/ => C/{1 0}/,C.comp/)
C,C.comp<=netL(net(W,W.comp X,X.comp) net(Y,Y.comp Z,Z.comp)))

bignetb forms a half oscillation pipeline component network of.

3.9.7.3. Figure 3.30c network expression: finer granularity of closure

bignetc of Figure 3.30c composed with three references to **netL**. The expression of **bignetc** inherits its locality structure and closure relations from the references to **netL**.

The dependency expression

(bignetc(W X Y Z => C)
C<=netL(netL(W X) netL(Y Z)))

The inheritance expanded network expression.

(bignetc(W/{1 0}/,W.comp/ X/{1 0}/,X.comp/
Y/{1 0}/,Y.comp/ Z/{1 0}/,Z.comp/ => C/{1 0}/,C.comp/)
C,C.comp<=netL(netL(W,W.comp X,X.comp) netL(Y,Y.comp Z,Z.comp)))

bignetc contains one oscillation network bounded by half oscillations.

3.9.7.4. Figure 3.30d network expression: integrating the link

The **"all of"** rank of a cross association search can serve as the enable behavior of a link. This can save some primitive behaviors with the tradeoff of increasing the inputs of other primitive behaviors. Whether such a tradeoff is useful depends on the specifics of a network. **bignetd** is composed from component network **net2** that contains the alternatively structured links.

In the expression of **net2** below ~C.close is applied to the cross association rank of **"all of"** behaviors that are determining the differentness of locality C. The rank of **"all of"** behaviors does double duty as interaction behaviors and as closure enable behaviors.


```
(net2(A/{1 0}/,AB.comp/  B/{1 0}/,AB.comp/ => C/{1 0}/,C.comp/)
        (allofrank/{3 2 1 0}/)
        [allofrank/{3<=[A/1  B/1]
                    2<=[A/1  B/0]
                    1<=[A/0  B/1]
                    0<=[A/0  B/0]  }  ~C.comp]
        AB.comp<=?C/{1<={allofrank/1  allofrank/2}
                     0<={allofrank/0  allofrank/3} } )
```

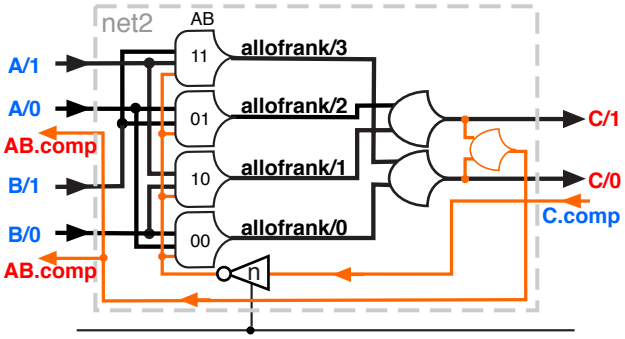


Figure 3.31. net2 network.

bignetd composed from three references to **net2**.

The dependency expression

```
(bignetd(W  X  Y  Z  => C)
            C<=net2(net2(W X) net2(Y Z))  )
```

The inheritance expanded network expression.

```
 (bignetd(W/{1 0}/,W.comp/  X/{1 0}/,X.comp/
             Y/{1 0}/,Y.comp/  Z/{1 0}/,Z.comp/  => C/{1 0}/,C.comp/)
        C,C.comp<=net2(net2(W,W.comp X,X.comp) net2(Y,Y.comp Z,Z.comp)) )
```

bignetd contains one oscillation network bounded by half oscillations.

3.9.7.5. Figure 3.30e network expression; finest granularity of closure

In Figure 3.30e the completeness and enable behaviors of the link are more finely integrated into the constant network locality by locality forming the longest pipeline version of the network with length of pipeline being characterized by the number of oscillation networks from input presentation to output assertion.

Each component **net3** network contains two links and one complete oscillation network bounded by half oscillations.


```
(net3(A/{1 0}/,in.comp/  B/{1 0}/,in.comp/ => C/{1 0}/,C.comp/)
        (allofrank/{3 2 1 0}/,allofrank.close/)
        in.comp<=?[allofrank/{3<=[A/1  B/1]
                              2<=[A/1  B/0]
                              1<=[A/0  B/1]
                              0<=[A/0  B/0] }  ~allofrank.comp]
        allofrank.comp<=?[C/{1<={allofrank/1 allofrank/2}
                              0<={allofrank/0 [allofrank/3} } ~C.comp] )
```

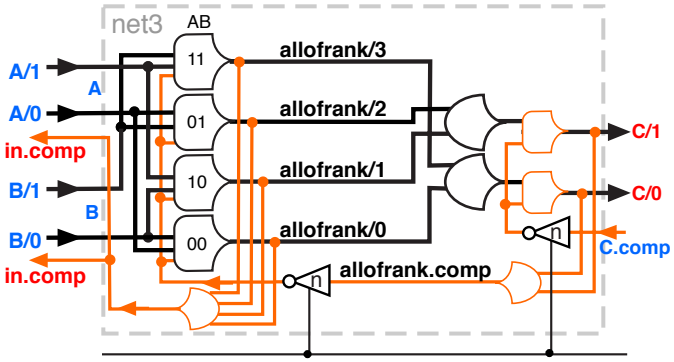


Figure 3.32. net3 network.

bignete is composed with three references to **net3**:
The dependency expression.

```
(bignete(W  X  Y  Z  => C)
              C<=net3(net3(W X) net3(Y Z))  )
```

The inheritance expanded network expression.

```
 (bignete(W/{1 0}/,W.comp/  X/{1 0}/,X.comp/
              Y/{1 0}/,Y.comp/  Z/{1 0}/,Z.comp/  => C/{1 0}/,C.comp/)
        C,C.comp<=net3(net3(W,W.comp X,X.comp) net3(Y,Y.comp Z,Z.comp))  )
```

bignete contains four oscillation networks two of which are concurrent. The network is bounded by half oscillations.

3.9.7.6. Webs of singularly appreciable transition completeness

The counter flowing networks form a web of singularity appreciation. In Figure 3.30c the counter flowing networks intersect through singularly appreciable component network outputs. In Figure 3.30e the counter flowing networks intersect through every locality, each an appreciable singularity of transition completeness which is dependent on other localities and has other localities dependent on it, forming a finer granularity of intersection.

3.9.7.7. Still a constant network

The closure network applied to a constant network does not affect the interaction behavior of the constant network. All five versions of the constant network and counter flowing closure networks in Figure 3.30 deliver the same interaction behavior. The escapement behavior of the component oscillation networks, affects the granularity of instance of interaction time and the throughput performance of the pipeline network.

3.10. INTERLUDE: The pipeline network

A pipeline network is a composition of linked oscillation networks bounded by half oscillations which forms a structure of interpenetrating counter-flowing networks with interaction transition wavefronts flowing through the pipeline network in the direction of interaction and with closure transitions flowing through the closure network counter to the direction of interaction flow. The counter flowing networks intersect through localities with links that coordinate both counter directional flows with singularly appreciable common transition completeness relations.

The constant network within the pipeline network, regardless of the closure structure encompassing it, remains the determiner of interaction behavior. Its behavior remains constant, always asserting the same output for the same presented input and it continues to fulfill the completeness criterion.

3.10.1. The exposed binding portal and the environment

Even though the component oscillation networks are spontaneously striving to oscillate a pipeline network as a whole remains dependent on the environment beyond its exposed binding portal to present monotonically transitioning input differentnesses, to provide liveness and variability of behavior.

The responsible environment
The environment still has an imposed requirement for monotonically transitioning the input to the pipeline network. Now an **input completeness half oscillation of the pipeline network determines the appropriate delay between transitions of its input presentation.** The responsibility of the environment is still to honor the closure of the input half oscillation of the pipeline network and if fulfilled the monotonic transitioning requirements for all the component oscillation networks of the pipeline network and their component primitive behaviors are also fulfilled.

From this point on all networks will be pipeline networks with counter flowing interaction and closure networks bounded by half oscillations.

3.10.2. Pipeline network interaction space

The constant network within the pipeline network remains the interaction space. The encompassment with a closure network does not alter the constant network and its interaction behavior.

3.10.3. Pipeline network interaction time

The flow of a wavefront of transition to **D** completeness from the input of the pipeline network to its output followed by a wavefront of transition to completely **N** marks one instance of interaction time for the pipeline network as a whole bounded by its binding portal and framed by transitions to completely **N** wavefronts.

There can be multiple wavefronts of transition to **D** completeness each followed by a wavefront of transition to completely **N** simultaneously flowing through stages of a pipeline network (section 3.9.6.4 and Appendix D). The oscillation networks marking instances of interaction time as the wavefronts of transition flow through them all cycle out of phase with each other and there is nothing stable about their phase relations. The internal behavior of a flowing pipeline network is an incoherence of fully coordinated but non-synchronous transitioning. There is no means of sampling an instant of stable analytically meaningful transition behavior across a flowing pipeline network. Only the closure network can appreciate as a singular wholeness the dynamic flow of wavefronts of transition through the dependency relations of the pipeline network.

The only singularly appreciable event differentiating instances of interaction for the pipeline network is the transition of the pipeline network output to **D** completeness followed by its transition to completely **N** marking one instance of interaction for the pipeline network as a whole.

3.10.4. Nothing new.

A pipeline network is still just a network of dependency relations among primitive association behaviors. The pipeline network emerges from the oscillation network which emerges from the completeness criterion fulfilling constant network which emerges from the primitive interaction behaviors. Each stage of emergence is just a particular composition of primitive association behaviors defined in section 3.2 which, themselves, emerged from the freely associating persistences and their interacting conditions in the shaking bag of Chapter 2.

3.10.5. No external metrics

There is no coherent reference frame or external metric of space or of time relative to a pipeline network and trying to impose an external metric onto the pipeline network contributes nothing to either the understanding of or the effective realization of the pipeline network.

3.10.6. The transcendental view

If one wishes, the closure network of a pipeline network can be removed leaving the raw constant network which can still be characterized as a single step mapping behavior transcending and ignoring its internal structure with its concurrent relations and ignoring the emergent behavior of the pipeline network.

3.11. The autonomous pipeline network: self control

An input/completeness half oscillation, instead of indicating when transitioned input can be accepted, can use its closure to form a completely presented input. A pipeline network can *auto produce* its own input. An output/enable half oscillation instead waiting for downstream closure

of the completely formed output wavefront can immediately determine it own completeness of output and close with itself to effectively *auto consume* its own output wavefront,.

Before this point network behavior has been entirely dependent on presented input from an external environment beyond the exposed binding portal of the network for liveness, for wavefront flow. The autonomous pipeline network now transitioning its own input presentation in its own time producing its own wavefronts is autonomously alive, is not dependent on any behavior external to the pipeline network. The pipeline network becomes isolated from the environment behaving independently producing its own wavefronts (liveness) in its own intrinsic time (rate of presentation) in its own intrinsic space (the network) as illustrated in Figure 3.33.

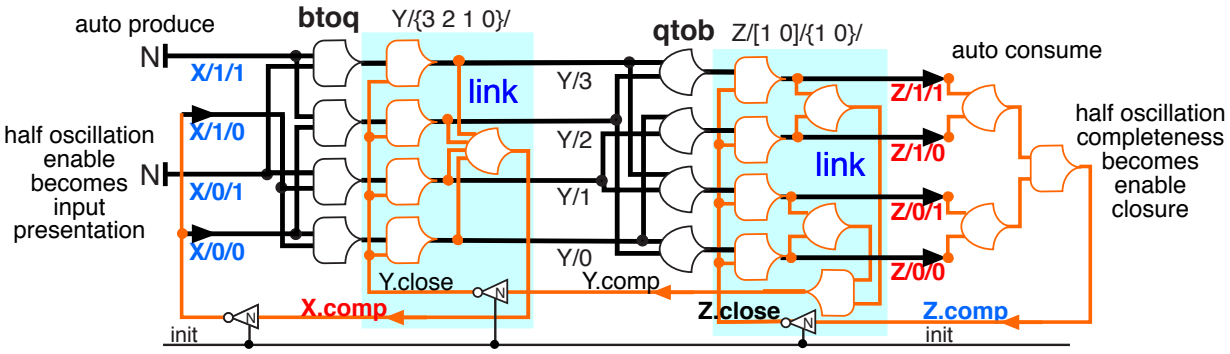


Figure 3.33. Pipeline network with auto produce and auto consume.

The expression of the network, with no activating and guiding reference from outside the network, has no binding portal and no reference name. The network is autonomous, continually alive and entirely on its own.

```
(   (   =>   )
      (Y  X  Z)
      Z<=[qtob(Y)  ~?Z]          /* auto consume */
      Y<=btoq(X/[1/{1<=N         /* auto produce  */
                    0<=~?Y }
                 0/{1<=N
                    0<=~?Y } ]  )  )
```

Z is dependent through **qtob** on **Y** and the converted completeness of **Z**. **Y** is dependent through **btoq** on **X**. **X/1/1** is constant /**N**. **X/1/0** is dependent on the converted completeness of **Y**. **X/0/1** is constant /**N**. **X/0/0** is dependent on the converted completeness of **Y**. Localities **X**, **Y** and **Z** inherit their locality structure and closure relations from **btoq** and **qtob**. There are no binding portal localities and there is no reference name. Hence there is no binding locality directional coloring in the expression. The expression cannot be referenced and copied and does not return a referencable result.

The network is continually presented with X/00 and will cycle indefinitely interacting X/00. The network is autonomously behaving and free of the environment but it has lost variability of behavior.

3.12. The embedded pipeline network: the passive environment

An autonomous pipeline network, embedded in an environment, can take full responsibility for its own behavior and still relate to the environment for variability of behavior by sensing the environment at its own rate through a **sampling portal** and imposing output behavior on the environment at its own rate through an **imposition portal**. The environment is still a source of variability of behavior for the network but it is no longer a source of liveness and time for the network. The environment is no longer in control of the network through an exposed binding portal.

The sampling portal and the imposition portal do not form an exposed binding portal. With an exposed binding portal the environment which references the network by name and binds external localities to the binding portal localities is in control of the network which passively and dependently waits on a presentation to its binding portal from the external environment.

With the sampling portal and imposition portal the embedded network is in complete control of sampling the environment and of imposing its output on the environment. The environment is indifferently passive neither waiting on nor dependent on the embedded network.

The network is now in complete control referencing the environment rather than the environment being in complete control referencing the network. The spontaneously behaving differentness of the network is manipulating and appreciating the passive differentness, information, of the external environment.

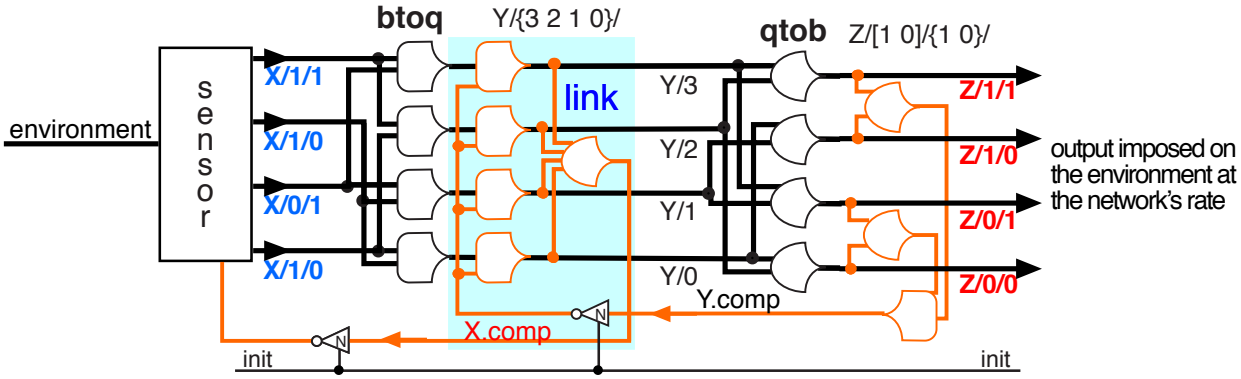


Figure 3.34. A pipeline senses the environment and imposes on the environment.

```
(   (   =>   )
      (Y  Z  sensor)
      imposedoutput<=Z<=qtob([Y  ~?Z])
      Y<=btoq(X<=[sensor ~?Y]  )  )
```


imposedoutput is dependent on **Z** which is dependent through **qtob** on **Y** and the converted completeness of **Z**. **Y** is dependent through **btoq** on **X** which is dependent on **sensor** and the converted completeness of **Y**.

3.13. INTERLUDE: The autonomous pipeline network

From the pipeline network with its half oscillation boundaries emerges the autonomous network in complete control of its wavefront flow. Closure has finally rendered the constant network independent of its environment. With the oscillation network closure first removed the requirement of the environment to wait an appropriate interval between presentation transitions supplying a time referent to the network. With auto produce and auto consume closure has further removed the requirement of the environment to monotonically transition input presentations supplying liveness to the network. The network is no longer anchored to an agency in the environment for the presentation of its input and is now free to move through the environment as a more complex and capable persistence continually interacting with the environment itself (see Chapter 2).

The network is finally in complete control of its own time and liveness on its own behavioral merits. It is not dependent on any extrinsic causative agencies or behaviors such as a mathematician with a pencil or an imposed clock interval or a responsible environment but is its own causative agency continually flowing its own wavefronts. The primitive behaviors passively subordinate to an external environment have collectively contrived to combine their external environments to achieve autonomy and independence.

While the network still relates to an external environment for its variability of behavior the network is in complete control of the relation. Control has migrated from the environment side of an exposed binding portal to the network side with sampling and imposition portals. The exposed binding portal has disappeared. The environment is now passive in relation to a determining network. Quandary 5 of section 3.6 is resolved. The network now expressing its own liveness and time and in complete control of the acquisition of its variability of behavior becomes an active explorer rather than a passive servant.

The no longer responsible environment
The isolated pipeline network now **creates its own monotonically transitioning presentation with the appropriate delay.** There is no longer any responsibility imposed on the environment. The constant network is now in complete control of itself autonomously behaving within an indifferent passive environment.

3.13.1. A complex persistence

Consider at this point that the passive environment might be a pure condition expression and that the autonomous network might be a complex persistence within it. That the sensor might be sensitive to specific differentness conditions within the environment. That the network enables the sensor and patiently waits for its sensor to encounter a specific condition which initiates a wavefront through the network causing a network behavior that affects the environment. Think proteins in cytoplasm.

3.13.2. Autonomous pipeline network space and time

The autonomous pipeline network is its own space of interaction. The monotonic transitions between **D** completeness and completely **N** auto produced to the input of the network are the ticks and tocks of its own interaction time. The consequent wavefront of each input flowing through the pipeline is one instance of interaction time delivering its own ticks and tocks to the component oscillation networks within the autonomous pipeline network.

3.13.3. Still nothing new

An autonomous pipeline network is still just a network of dependency relations among primitive association behaviors (section 3.10.4)

A newly introduced element here is the sensor which is a transducer sensitive to the environment which may or may not be realizable in terms of primitive interaction behaviors.

3.13.4. No external metrics

There is no coherent reference frame or metric of space or of time relative to an autonomous pipeline network and trying to impose an external metric onto the autonomous pipeline network contributes nothing to either the understanding of or the effective realization of an autonomous pipeline network.

3.13.5. The transcendental view

The closure structure can still be removed to characterize the encompassed raw constant network as a single step mapping behavior that still always provides the same asserted output for the same presented input but the raw network returns to complete dependency on the environment.

3.14. The journey

The last sentence of Chapter 1 declared the pursuit of *an accounting of interaction complete and sufficient in itself with no need of extrinsic support.* The journey began with primitive behaviors made sufficiently expressive and continued to the constant network of dependently related primitive behaviors which remains dependent for its liveness, its referent of time and its source of variability of behavior on presentations of input from an external environment which is in *complete control of the passive network*. The journey continued through the oscillation network, expressing its own referent of time, to the pipeline network with its bounding half oscillations that close on themselves expressing its own source of liveness. The autonomous network emerged *sufficiently expressive and in complete control of itself* taking from a passively indifferent external environment instead of receiving from a controlling external environment.

3.14.1. A first principle fulfilled

An autonomous pipeline network is *differentness spontaneously and dependently interacting and changing*(see Chapter 2 and section 2.5.5).

3.14.2. A journey not yet complete

The exposed binding portal is gone and the network is in complete control of its own behavior but there remains a dimension of expressivity still to be considered. The common denominator of this chapter is that every example network accepts an input which initiates a

wavefront that flows to the output and out of the network performing one isolated instance of interaction in one isolated instance of interaction time. The following wavefront of transition to completely **N** then erases the instance of interaction from the network isolating successive instances of interaction and allowing the network to be reused with another wavefront. The networks have no behavioral memory and no means of relating different instances of network interaction, different interaction wavefronts, in the context of the network even when the network is in complete control of itself (section 3.4.7.4). The next chapter addresses this expressional inadequacy.

Chapter 4:
Temporal differentiation

To this point temporal differentiation has manifested as interaction time. One flow of transition to **D** completeness wavefront into through and out of a constant network followed by a transition to completely **N** wavefront erasing the interaction from the constant network marked one instance of interaction time in relation to the constant network (sections 3.5.2.2 and 3.4.7.4). A subsequent flow through the same constant network of transition to **D** completeness wavefront followed by a transition to completely **N** wavefront marked a different isolated instance of interaction time in relation to the same constant network. Instances of interaction time are differentiated in the context of the network but the constant network with no behavioral memory is not able to account its differentnesses of interaction time. Temporal differentness does not persist to be referenced within the context of the network precluding the constant network from representing a complete and coherent accounting of differentness spontaneously and dependently interacting and changing (section).

This chapter is about accounting the interaction of temporal differentnesses. Instances of interaction do not flow out of a network but are remembered and accounted flowing within the network which becomes a complete and coherent expression of interaction in contrast to being a fragment of interaction expression (section 3.4.6.5).

4.1. The ring network: boundless network, endless time

Associating the output/assertion half oscillation of a pipeline network directly to its input/presentation half oscillation forms a ring network. The pipeline network's input and output becomes completely internalized. The network becomes completely isolated. Interacting wavefronts no longer flow out of the network but they also no longer flow into the network. Whatever the network is doing it is completely accounting it.

4.1.1. The base pipeline

For this example the initial pipeline is the component pipeline networks **qtob** and **btoq**, of Figure 3.27, linked binary to binary with quaternary half oscillation boundaries, Figure 4.1.

The dependency expression.

```
(basepipe(X=> Z )
      Z<=btoq(qtob(X))     )
```

The inheritance expanded network expression.

```
(basepipe(X/{3 2 1 0}/,X.comp/ => Z/{3 2 1 0}/,Z.comp/ )
      Z,Z.comp<=btoq(qtob(X,X.comp))     )
```

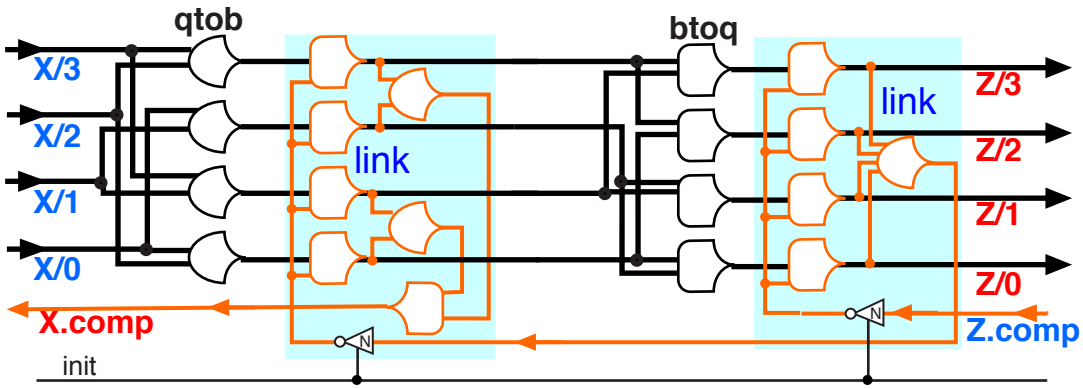
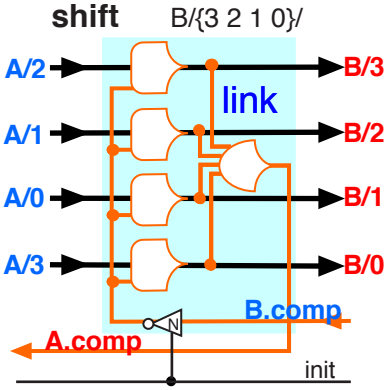



Figure 4.1. Base pipeline for the ring

4.1.2. Closing the pipeline

A ring network is a composition of linked oscillation networks which must include at least three oscillation networks to accommodate a transition to **D** completeness wavefront, a transition to completely **N** wavefront and one bubble (Appendix D). If output assertion half oscillation **basepipe/Z** is connected directly to input presentation half oscillation **basepipe/X** the resulting network has only two oscillation networks. So a pipeline component must be inserted between **Z** and **X** to form another oscillation network. The inserted pipeline component performs a shift of quaternary differentness.




```
(Shift(A/{3 2 1 0}/,A.comp/ =>
                B/{3 2 1 0}/,B.comp/)
A.comp<=?[B/{3<=A/2
          2<=A/1
          1<=A/0
          0<=A/3} ~B.comp]  )
```

Figure 4.2. The shift buffer pipeline component.

A.comp is dependent on the completeness of **B** which is dependent on shifted components of **A** which are dependent on **~B.comp**.

The dependency expression.

```
(   ( =>  )          (Z:3/{3 2 1 0}/)
     Z<= basepipe(shift(Z) ) )
```

Z initialized to **Z/3** is dependent through **basepipe** and through **shift** on **Z**. The dependency of **Z** on **Z** closes the ring. **Z** inherits locality structure and closure relations from **basepipe/Z** and from **shift/A**.

The inheritance expanded network expression.

```
(   ( =>  )          ( Z:3/{3 2 1 0}/,Z.comp/ )
     Z,Z.comp<= basepipe(shift(Z,Z.comp) )
```

The expression has no binding portal, no reference name, no boundary localities, nothing flows into or out of the ring, hence there is no binding portal directional coloring in the expression. With no boundary for

wavefronts to flow into the ring a **D** completeness wavefront and a completely **N** wavefront must be initialized in the ring. This is expressed with **(Z:3)** specifying that **Z/3** be initialized to **D** (Appendix E).

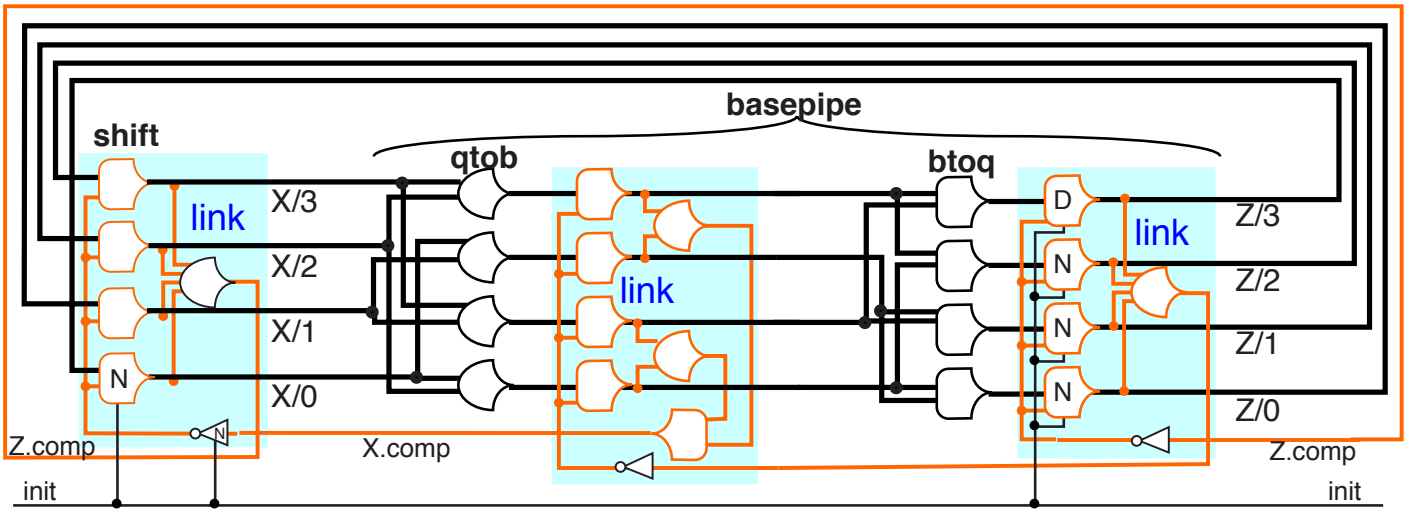


Figure 4.3. The ring network.

4.2. INTERLUDE: The ring network

Once instantiated the ring network is a completely self contained network providing its own liveness, time and variability of behavior. The initialized transition to **D** completeness wavefront followed by a transition to completely **N** wavefront autonomously, spontaneously and neverendingly flow around the ring at its own rate in its own space forever or until it fails from internal wear or external insult.

4.2.1. A new wholeness

The ring network is a wholeness of accounting representing a wholeness of expression as well as a wholeness of network (sections 3.4.6.5 and 3.4.10.8 and 3.5.2.4). The ring network now says everything there is to say about its interaction behavior.

4.2.1.1. The loss of singular appreciability

The ring network, with no exposed binding portal, no input or output, and with its wavefront continually flowing around the ring network never having flowed into the ring network and never flowing out of the ring network, has no singular referent to characterize a beginning of instance of interaction or instance of interaction time or an end of instance of interaction or instance of interaction time for the ring network as a whole. There is no output assertion boundary to always assert the same output for the same input presented to the equally nonexistent input presentation boundary.

There is no one transition to completeness that is privileged over any other transition to completeness in the context of the ring network as a whole. The ring network as a whole no longer has a singularly appreciable tick of time.

4.2.1.2. The incorporated external environment

The ring network with no referent of time itself is now managing the presentation of binding portal input for all of its component oscillation networks and hence determining instance of interaction and instance of interaction time for each component oscillation network. Each instance of interaction wavefront flows out of and is erased from each component oscillation network but each instance of interaction wavefront remains within the ring network flowing to a next interaction. The ring network has become an external environment

to its component oscillation networks but an external environment fully incorporated into the ring network as a whole.

4.2.1.3. The master of interaction time

The ring network wavefront flowing around the ring uses, unuses and reuses each component oscillation network over and over. The ring network is managing the interaction time of its component oscillation networks but nothing manages time for the ring network as a whole. The continually flowing ring network wavefront manifests independently flowing, self regulating time for its component oscillation networks. There is no singularly appreciable referent of interaction time for the flowing ring network wavefront but it is in complete control of the flow of interaction time nevertheless.

4.2.2. Still just a composition of primitive behaviors

A ring network is still just a network of dependency related primitive association behaviors that emerges from the pipeline network (section 3.10.4).

4.2.3. No metrics

There is no coherent reference frame or metric of space or of time relative to a ring network and trying to impose an external metric onto the ring network contributes nothing to either the understanding of or the effective realization of the ring network.

4.2.4. The end of the transcendental view: a boundary crossed

The ring network cannot be transcendentally characterized abstractly as a single step mapping behavior apart from its closure structure. Firstly, there is no mapping from a singularly appreciable input to a singularly appreciable output. Secondly, the ring network with its closure structure removed does not work. With the ring network the oscillation network becomes an essential un-transcendable unit of composition.

4.3. The source ring network: making time

A completely isolated ring network can extend beyond itself by linking the assertion boundary of any component oscillation network of the ring as an output of the ring network. Locality **S** is added and in Figure 4.4 as an assertion output dependent on **Z**.

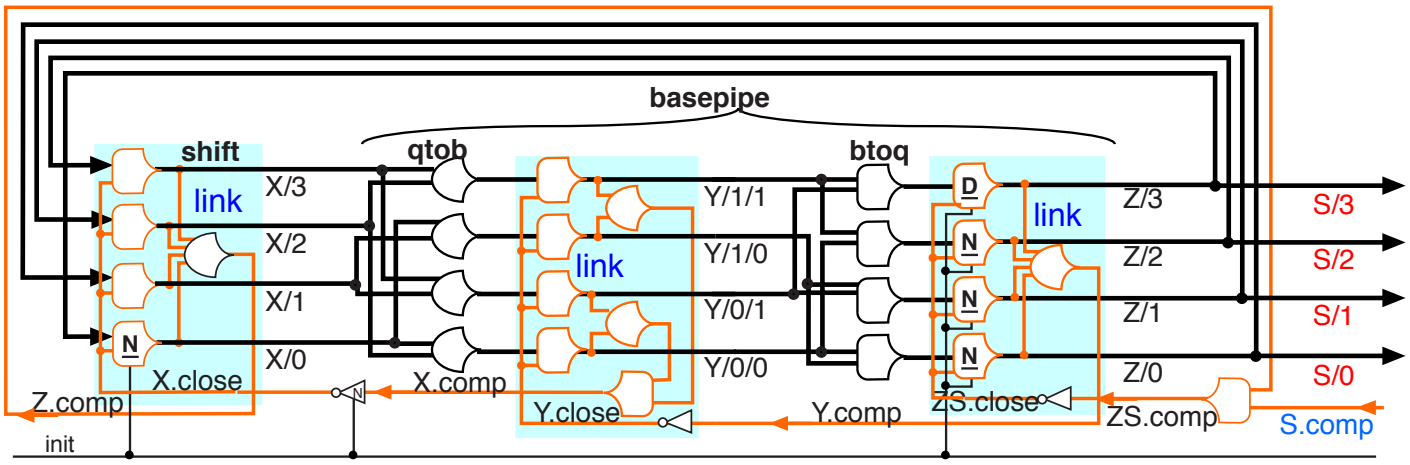


Figure 4.4. The source ring network.

Since the ring expression cannot be referenced because it has no reference name or binding portal the dependency expression of the ring expression is recapitulated for the source ring. For the source ring there is no presentation locality but there is an assertion locality to be referenced so the source ring expression invites reference with a half binding portal with reference name **source**.

The dependency expression.

(source(=> S) (Z:3)
Z<=S<=basepipe(shift(Z))

S is dependent on **Z** initialized to **Z/3** which is dependent through **basepipe** and through **shift** on **Z**. The dependency of **Z** on **Z** closes the ring. **Z** flows to two destinations **Z** and S. S and **Z** inherit locality structure and closure relations from **basepipe/Z** and **shift/A**.

The inheritance expanded network expression.

(source(=> S/{3 2 1 0}/,S.comp/) (Z:3/{3 2 1 0}/,Z.comp/)
Z,Z.comp<=S,S.comp<=basepipe(shift(Z,Z.comp))

The source ring can be referenced as:

reference nested	full portal reference	locality nested
(source( ))	**source(=> M)**	**M<=source( )**

through the input half oscillation of a pipeline network.

4.3.1. Source ring network behavior

The continually flowing wavefront of the source ring network produces successive wavefronts of transition monotonically transitioning between **D** completeness and completely **N** at S delivering discrete differentiated instances of interaction time as input to a referencing pipeline network.

The pipeline network is dependent on presentations monotonically transitioning between **D** completeness and completely **N** as a source of liveness, a source of variability of behavior and a source of time from beyond itself honoring its closure. Its beyond, however, is not an uncharacterized external environment but is fully characterized within the source ring network which as a whole is not dependent on anything. As far as the source ring is concerned it is entirely indifferent to the S portal making its continually flowing wavefront occasionally wait.

One might view the ring network "environment" as controlling the pipeline network with input presentations as an external environment does through an exposed binding portal but the ring network, the "environment", is not referencing the receiving pipeline network. The receiving pipeline network is referencing the ring network. The pipeline network initiates and is in complete control of the reference grabbing with its closure an instance of interaction and interaction time from the cornucopia wavefront of the ring network. This is similar to the sampling portal of the network of section 3.12 with its closure grabbing discrete instances of interaction from a passive and continuous external environment.

4.4. The pipeline ring network: from time to time

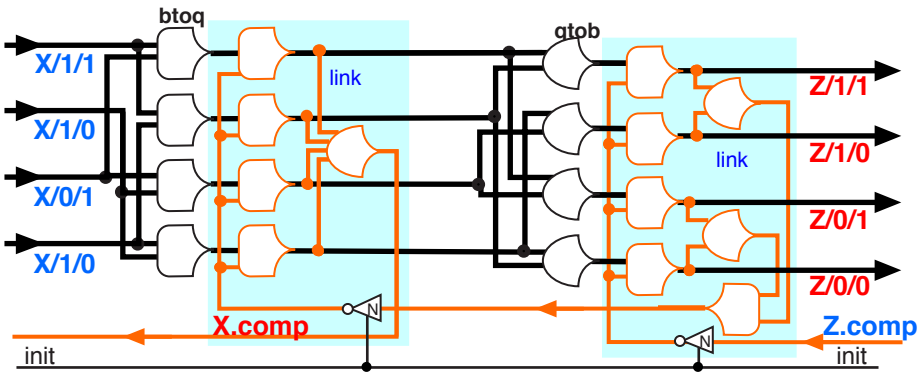
A ring network and a pipeline network sharing at least one oscillation network forms a **pipeline ring network** as in Figure 4.6. The example pipeline ring named **pipering** is composed in terms of a pipeline network **pipeline** with **btoq** and **qtob** connected quaternary to quaternary.

The dependency expression

(pipeline(X=> Z)
Z<=qtoq(btob(X)))

The inheritance expanded network expression:

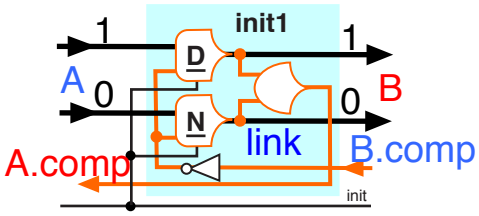
(pipeline (X/[1 0]/{1 0}/,X.comp/ => Z/[1 0]/{1 0}/,Z.comp/)
Z,Z.comp<=qtoq(btob(X,X.comp)))



Closing a ring through pipeline requires at least one more oscillation network and the initialization of a **D** completeness wavefront which requires two oscillation networks one initializing the **D** completeness wavefront followed by one initializing a completely **N** wavefront to block the flow of the **D** completeness wavefront during initialization (Appendix E).

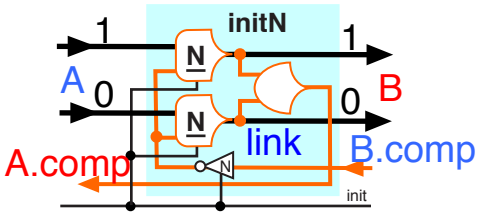
Pipeline component initializing to **1.**

```
(init1(A/{1 0}/,A.comp/ =>
                B:1/{1 0}/,B.comp/)
   A.comp <=?B<=[A  ~B.comp]  )
```



Pipeline component initializing to **N.**

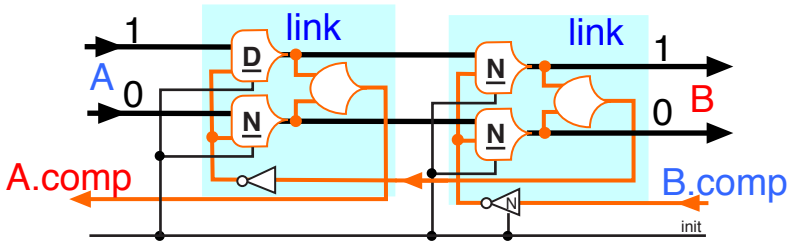
```
(initN(A/{1 0}/,A.comp/ =>
                B:N/{1 0}/,B.comp/)
   A.comp <=?B<=[A  ~B.comp]  )
```



The pipeline components are composed into a pipeline segment **pipeseg1** initializing a **D** completeness wavefront to **1** is defined.

Pipeline segment to inialize a **D** completeness wavefront.

```
(pipeseg1(A/{1 0}/,A.comp/ => B/{1 0}/,B.comp/)
      B,B.comp<=initN(init1(A,A.comp ))  )
```



The dependency expression.

```
(pipering(in => out)          (A)
      A<=out<=pipeline([pipeseg1(A)   in] ) )
```


The inheritance expanded network expression.

(pipering(in/{1 0}/,in.comp/ => out/{1 0}/,out.comp/) (A/{1 0}/,A.comp/)
[A,A.comp out,out.comp]<=pipeline([pipeseg1(A,A.comp) in,in.comp]))

To form the pipeline ring the components of the **in** and **out** localities split and associate separately. **in/1** and **out/1** connect to the ring. **in/0** is the network input and **out/0** is the network output. The syntax [A **out**]<=pipeline indicates that the output of **pipeline** associates to **A** and to **out** corresponding to the topmost order level of composition of the **pipeline** output locality **Z**. Locality **Z/1** associates to **A** and **Z/0** associates to **out**. This association relation is illustrated on the left of Figure 4.5.

The syntax <=pipeline([**A in**]) indicates that the input of **pipeline** associates from **A** and **in** corresponding to the topmost level order of composition of the **pipeline** input locality **X**. Locality **B** associates to **X/1** and locality **in** associates to **X/0**. This association relation is illustrated on the right of Figure 4.5.

The **"all of"** relations between the reference and the definition associate then the ordered terms within the **"all of"** relations associate.

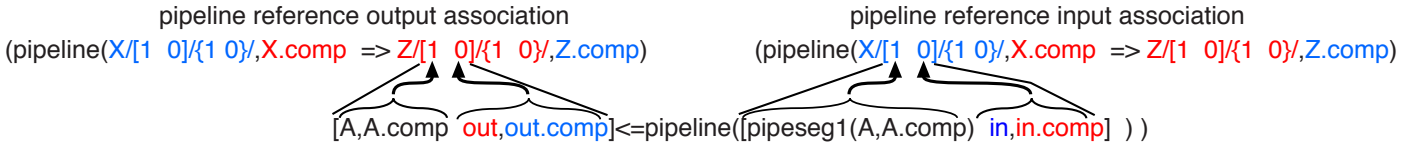


Figure 4.5. Binding portal associations for pipeline reference.

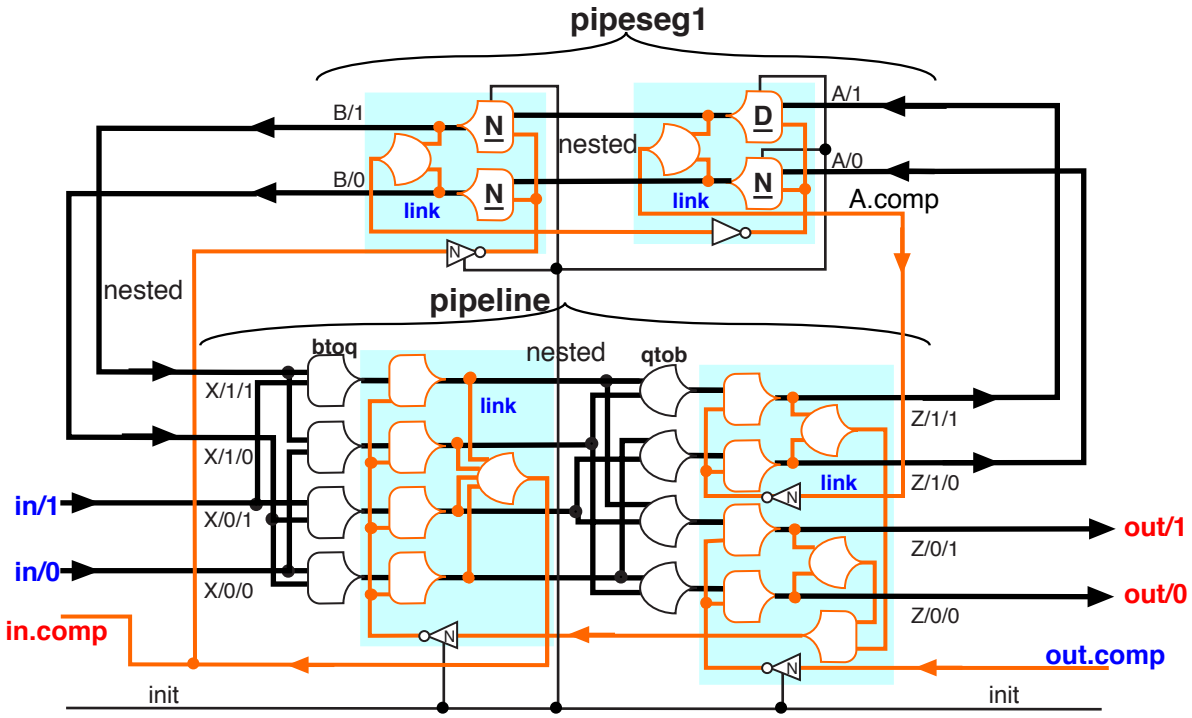


Figure 4.6. The pipeline ring network.

4.4.1. Pipeline ring behavior

The pipeline network has an exposed binding portal and just like the pipeline networks of the previous chapter receives presented input and delivers asserted output through its exposed binding portal. Each wavefront begun by the presented input **in** will interact with a wavefront from the ring network as it flows to the output **out** of the pipeline network. There has to be a first **D** completeness - completely **N** wavefront pair initialized in the ring network that will interact with the first transition to **D** completeness - completely **N** wavefront pair presented to the pipeline network. In this case the **D** completeness wavefront is initialized in component network **pipeseg1** and the completely **N** wavefront is initialized in locality **Z/1**. The first wavefront pair entering the pipeline after initialization interacts with the initialized wavefront pair in the ring network. This first instance of interaction produces the first **D** completeness wavefront to enter the ring network which is a memory of the first pipeline instance of interaction. The following completely **N** wavefront overwrites the **D** completeness wavefront in the pipeline network but the **D** completeness wavefront in the ring is not overwritten and lingers in the siding of the ring. The second **D** completeness wavefront entering the pipeline network interacts with this first pipeline instance of **D** completeness interaction wavefront lingering in the ring network. The third **D** completeness wavefront to enter the pipeline network will interact with the memory of the second pipeline **D** completeness instance of interaction lingering in the ring network. And so on.

The ring network coupled to the pipeline network with its wavefront flowing backward in pipeline space and forward in pipeline time enables a pipeline network to retain an interaction result wavefront and associate it with and interact with a future input wavefront of the same pipeline network. A present instance of interaction wavefront interacts with a past instance of interaction wavefront. The network now has an internal memory establishing an interaction dependency relation across different instances of interaction time. Temporal memory has emerged in relation to the network. Such a relation was not possible in the networks of Chapter 3.

4.4.2. No longer constant

The pipeline ring network is the first example of a network with an exposed binding portal that is not constant. The pipeline ring network as a whole no longer asserts the same completeness of output for the same completeness of presented input.

4.5. INTERLUDE: A collision of expression regimes

The pipeline ring network represents a collision of expression regimes. The environment beyond the exposed binding portal is not in complete control of the network behavior because of the ring and the ring within the network is not in complete control of the network behavior because of the exposed binding portal. There are two sources of wavefront flow, of liveness and of time. The colliding wavefront sources cooperate and coordinate through a shared oscillation network but cooperation is not an arrow in the quiver of either regime.

A Tth wavefront flows into the pipeline network through the binding portal and a Tth wavefront flows out of the pipeline network. The fact that there is a ring network coupled to the pipeline network associating and interacting different instances of interaction time inside the pipeline network is not visible to the exposed binding portal. The external environment presenting to the exposed binding portal, however, must take into account this invisible influence on the interaction behavior of the pipeline which varies from presentation to presentation. It is a new kind of component object with not just a mapping of differentness context to considered but also with a temporal context to be taken into consideration.

4.5.1. The environment expression regime

It is this influence on behavior not visible through the exposed binding portal (side effect) that concerns functional programming which strives to maintain, at all cost, the constancy of the network expression (stay functional) and for the environment presentation through the exposed binding portal to be in complete control of the interaction (referential transparency), i.e. all sources of wavefront flow are presented from the external environment and there are no sources of wavefront flow, no expressions of liveness or of time, from within the network itself such as a memory relating different instances of interaction. The functional view of interaction cannot get out of Chapter 3.

4.5.2. The network expression regime

The goal of this narrative is to completely characterize interaction in terms of a network of dependency relations among primitive behaviors in complete control of itself, i.e that all sources of wavefront flow are from within the network expression and there are no sources of wavefront flow, no expressions of liveness or of time, presented from outside the network expression.

4.5.3. The exposed binding portal

The difference between these two expression regimes is the presence of or the absence of a binding portal exposed to an environment unaccounted by the network. If a network possesses a single exposed binding portal then the entire network and all of the internal sources of wavefront flow, the rings, will patiently wait on the presentation from the exposed binding portal, i.e. the external unaccounted environment remains in complete control of the liveness and temporal behavior of the network. With the pipeline ring network the external environment remains in control.

An exposed binding portal is a leak of accountability in the context of the network (section 3.5.2.4). The network cannot encompass a complete accounting of interaction if it remains dependent on an exposed binding portal. The unaccounted environment can be an arbitrarily complex interaction (a human?) just beyond the exposed binding portal. The only way for a network to completely account its interaction is for the network to not have any binding portals exposed to an unaccounted external environment. The network must be in total control of all relations within the network and with any external environment. Assuming a responsible external environment is not an option for the network.

4.5.4. Still nothing new

The pipeline ring network is still just a network of dependency relations among primitive behaviors.Nothing new has been introduced.

4.5.5. No metrics

There is no coherent reference frame or metric of space or of time relative to a pipeline ring network as a whole and trying to impose an external metric onto the pipeline ring network contributes nothing to either the understanding of or the effective realization of the pipeline ring network.

4.6. Removing the exposed binding portal

The exposed biding portal of Figure 4.6 can be removed and its boundary incorporated into the network expression by closing the pipeline into a ring to form a network of coupled rings or by the half oscillations of the pipeline closing on themselves forming an autonomous pipeline network or by linking the pipeline network to ring networks forming a network of coupled and linked rings.

4.6.1. The coupled ring network

The pipeline component network of the **pipering** network can be closed by associating its output to its input through **pipeseg1** forming a network of two rings coupled through a shared oscillation network as in

Figure 4.7. The new ring still has to be at least three oscillation networks and must have an initialized **D** completeness and completely **N** wavefront pair so the **pipeseg1** pipeline component network is needed again.

```
( ( => ) (out:0/{1 0}/,out.comp/)
    out,out.comp<=pipering(pipeseg1(out,out.comp) ) )
```

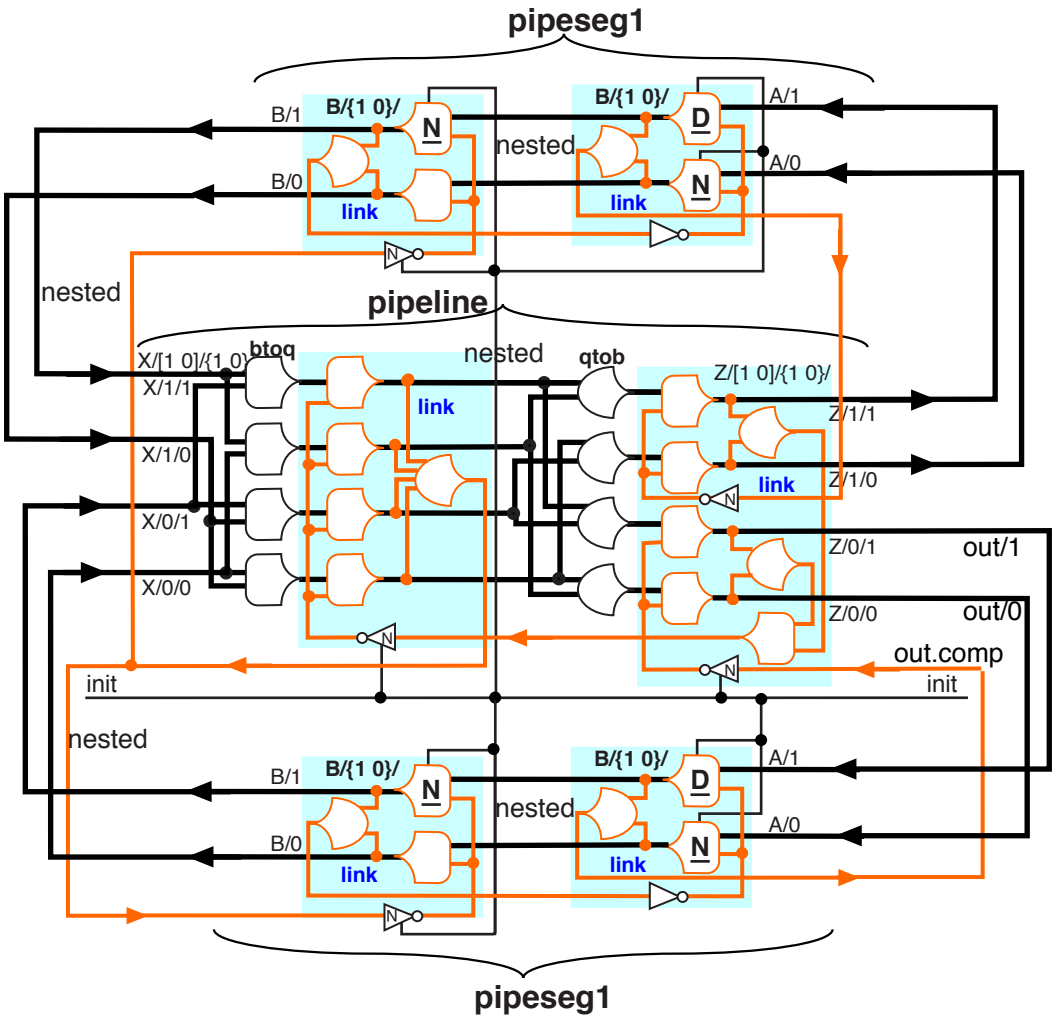


Figure 4.7. The network of two coupled rings.

Each ring is an internal source of wavefronts and of time which times are coordinated through a shared oscillation network. Neither ring cares that its wavefront might be delayed by the other ring. Each oscillation network including the shared oscillation network mark an instance of interaction time as the ring wavefronts flow through them. But the interaction network as a whole does not mark instances of interaction time. The two wavefronts form one single never ending instance of interaction time.

4.6.2. The autonomous pipeline network

In Figure 4.8 the exposed binding portal half oscillations of the component pipeline network close on themselves forming an autonomous pipeline network. The pipeline ring network becomes self determined like the network of Figure 3.34.


```
(   (   =>   )          ( sensor/{1  0}/,sensor.comp   out/{1 0}/,out.comp)
                  out.comp<=?out<=pipering(sensor,sensor.comp)   )
```

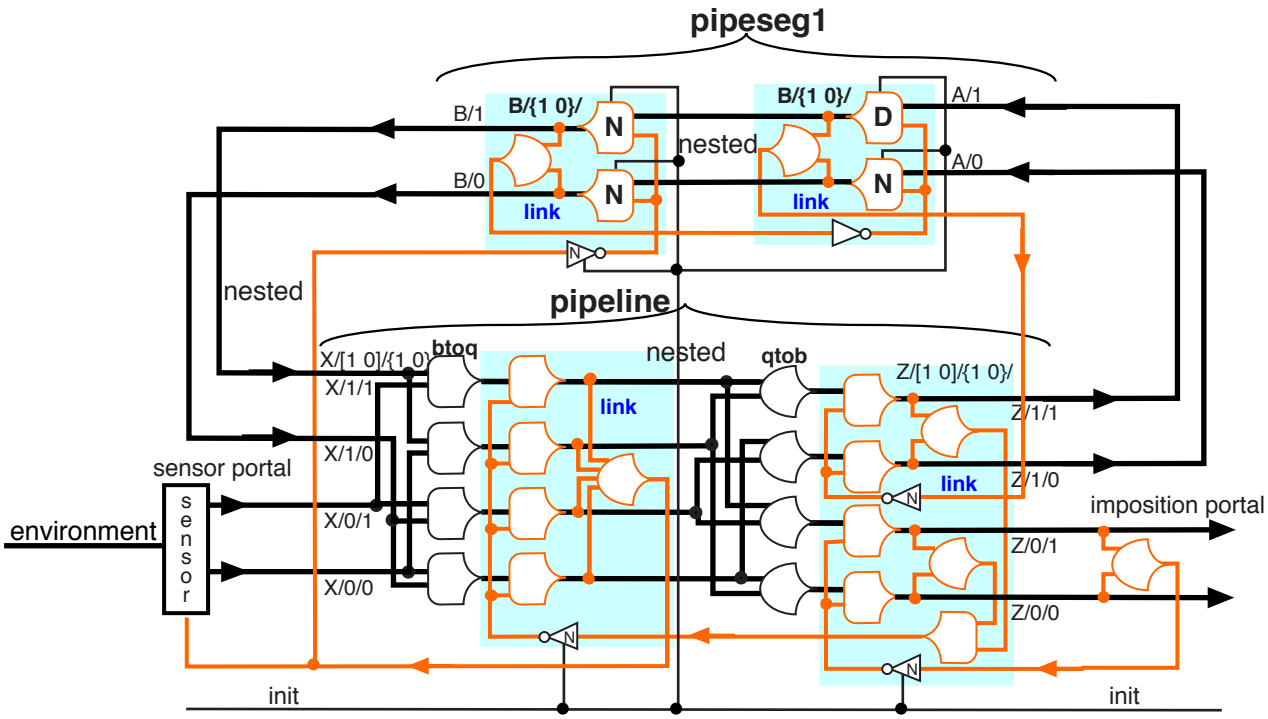
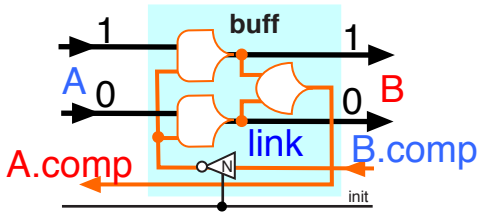


Figure 4.8. The pipeline network of pipering with auto produce and auto consume.

4.6.3. The network of coupled and linked rings

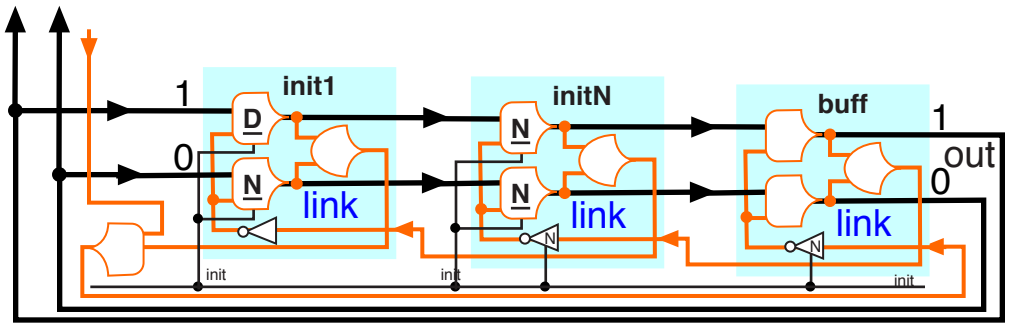
The binding portal half oscillations of the component pipeline network are linked to rings in Figure 4.9 which supply the input and receive the output of the pipeline. One more pipeline component **buff** that does not initialize is needed to form the three oscillation ring networks.

```
(buff(A/{1 0}/,A.comp/ => B/{1 0}/,B.comp/)
          A.comp<=?B<=[A  ~Bcomp]  )
```



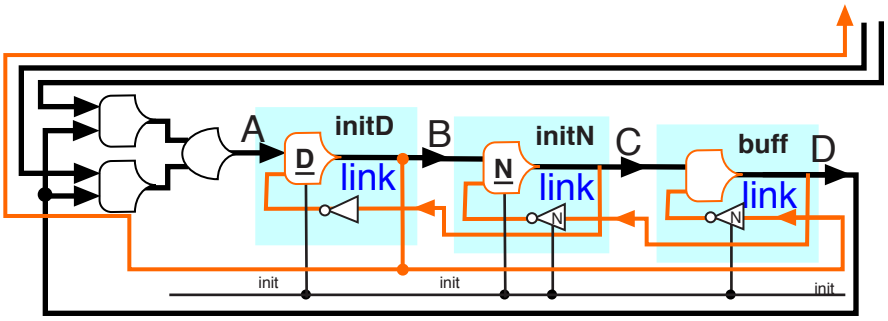
Network **sourcering** is expressed using **buff** and **init1** and **initN** from section 4.4.

```
(sourcering(   =>  out/{1  0}/,out.comp)          (Z/{1  0}/,Z.close)
      out<=Z,Z.close<=initN(init1(buff(Z,[Z.close  out.comp])))   )
```

Network **sinkring** is a single rail ring that sinks the output wavefront.

(sinkring(in/{1 0},in.comp/ =>) (A/ B/:D C/:N D/)
A<=?[in D] [in.comp B]<=[A ~C]
C<=[B ~D] D<=[C ~B]



The pipeline ring network with input and output ring caps.

```
(  (  =>  )
     sinkring(pipering(soucering( ) ) )  )
```

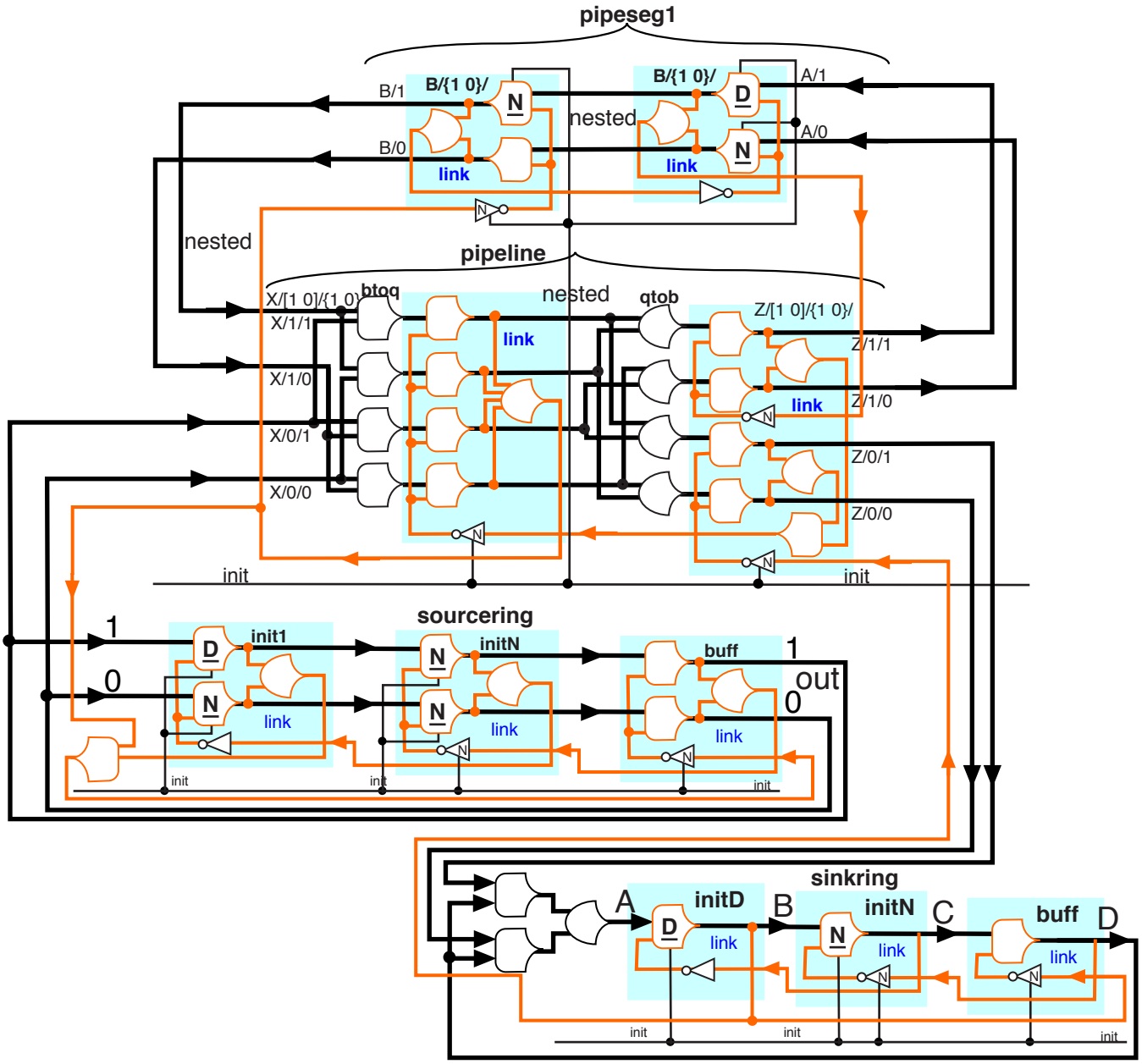


Figure 4.9. The pipeline network of pipering linked to rings.

The component pipeline network is coupled to a ring through a shared oscillation network and is linked to an input ring and an output ring. All sources of wavefront flow are within the network.

4.7. INTERLUDE: The self determined network

Rings coupled through shared oscillation networks, rings linked through shared pipelines and imposition portals and sampling portals support indefinitely complex self determined network expressions. In the network of Figure 4.9 there are three different wavefronts in three different rings flowing and coordinating

their flow in relation to each other. Consider the possibility of a trillion mutually coordinating and interacting wavefronts flowing in a network of a trillion linked and coupled rings.

4.7.1. Nothing in control

With no exposed binding portal there is no singular authority that determines liveness, time and wholeness for the self determined network (sections 3.5.2.4 and 3.4.6.5). There is only the network itself and its intrinsic wavefronts. A self determined network is egalitarian (section 4.2.1). No boundary is more authoritative than any other boundary. Everything is equally subordinate to everything else. Everything is equally dependent on everything else. Everything is equally determinative. There is nothing referentially central. There is nothing referentially global. There is no intrinsic authority. There is only wholeness of network and wholeness of expression. There is nothing that references a self determined network. A self determined network does not report to anything else. A self determined network is a complete accounting of interaction within itself free to move through and relate to an external environment and to interact with other self determined networks.

4.7.2. No appreciable singularity of network behavior

With no exposed binding portal there is no singularly appreciable referent, no tick, of instance of interaction or instance of interaction time in relation to the self determined network as a whole. There is only the incoherent transitioning cacophony of its component oscillation networks (sections 3.5.2.3 and 3.10.3). The only agency capable of effectively appreciating the cacophony of interaction behavior within a self determined network is the counter flowing closure network (section 3.9.7 and 3.10.3) which is also an integral part of the self determined network managing the cacophony of inchoherent behavior in terms of the coherency of dependency completeness relations.

4.7.3. Dimensions of differentiation

Considering computation in terms of differentness reveals three interpenetrating dimensions of differentness: differentness of condition (Chapter 2) within differentness of association (Chapter 3) within differentness of time (Chapter 4). Differentness of condition and differentness of association, interpenetrate, collaborate and mutually extend (Chapter 5). The differentness of wavefronts of change flowing through the first two dimensions forms the third dimension of temporal differentness of interaction (Chapter 4) which is not persistently referencable as are the first two dimensions.

4.7.4. The temporal dimension of differentiation

While flowing wavefronts of transition use, unuse and reuse the differentness of each component oscillation network over and over extending their expression of differentness through interaction time there is nothing that similarly reuses and extends the differentness of interaction time over and over. Differentness of interaction time can never be reused because each instance of differentnesses of interaction time is irretrievably erased, irreferencibly unused, by the following transition to completely **N** wavefront.

Interaction time is ephemerally self limiting but it is also indefinitely self extending (sections 3.4.7.4 and 3.5.2 and 3.5.2.1). Transition to **D** completeness wavefronts interact producing subsequent transition to **D** completeness wavefronts which interact producing subsequent transition to **D** completeness wavefronts and so on. **D** completeness wavefronts flowing progressively and indefinitely represent the flow of time for a self determined network. A **D** completeness wavefront trails behind itself **D** completeness bubbles (section 3.9.6.4 and Appendix D). It is the **D** completeness bubbles that the following wavefront of transition to completely **N** wavefront erases. A transition to completely **N** wavefront cannot overtake and erase a transition to **D** completeness wavefront. Similarly a transition to **D** completeness wavefront can only flow into completely **N** bubbles and cannot overtake and compromise a completely **N** wavefront.

In a self determined network **D** completeness wavefronts form narrow bands of continually progressing transition behavior followed by the transition to completely **N** wavefronts. Each ring network supports one bandlette of **D** completeness wavefront. These narrow bands of **D** completeness transition behavior form the dimension of temporal differentiation within which differentnesses of interaction time can associate and interact. Interaction time is the last but indefinitely extending dimension of differentiation and interaction. There can be just so many differentness conditions and an association network can be only so big but there is no limit to the extension of time.

4.8. The LFSR network: interacting differentnesses of time

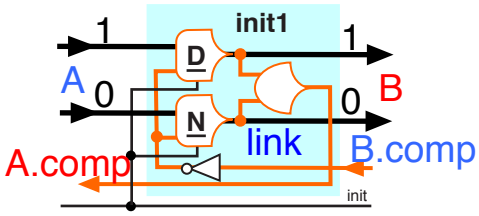
The LFSR (Linear Feedback Shift Register) network of Figure 4.10 is a network of 5 rings with 6 independently flowing wavefronts coupled through a structure of shared oscillation networks. Each wavefront represents a different instance of interaction time with 5 instances interacting through a shared pipeline network.

The LFSR network is constructed with pipeline component networks **pipe1** which initializes a wavefront to **1**, **pipe0** which initializes a wavefront to **0**, **pipeN** which initializes a wavefront to completely **N** and a pipeline of **XOR** networks.

4.8.1. Pipeline component networks for LFSR

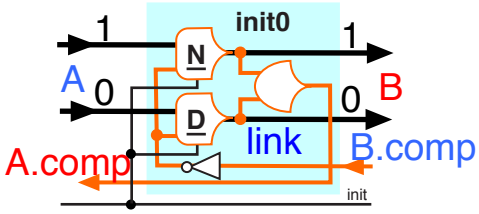
Pipeline component initializing to **1**

```
(init1(A/{1 0}/,A.comp/ =>B:1/{1 0}/,B.comp/)
    A.comp<=?B<=[A  ~B.comp]  )
```



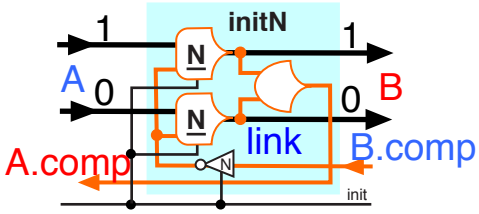
Pipeline component initializing to **0**

```
(init0(A/{1 0}/,A.comp/ =>B:0/{1 0}/,B.comp/)
    A.comp<=?B<=[A  ~B.comp]  )
```



Pipeline component initializing to **N**

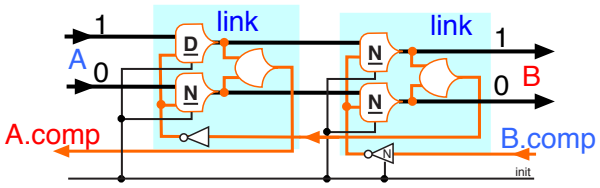
```
(initN(A/{1 0}/,A.comp/ =>B:N/{1 0}/,B.comp/)
    A.comp<=?B<=[A  ~B.comp]  )
```



4.8.2. Initializing pipeline segment networks for LFSR

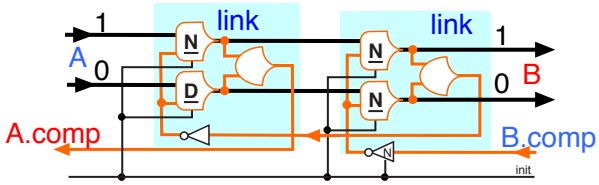
pipe1 pipeline segment

```
(pipe1(A/{1 0}/,A.comp/ => B/{1 0}/,B.comp/)
    B,B.comp<=initN(init1(A,A.comp))  )
```



pipe0 pipeline segment

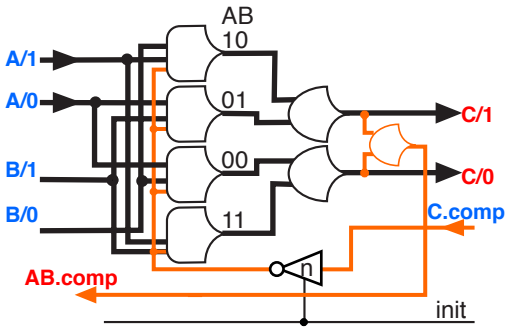
(pipe0(A/{1 0}/,A.comp/ => B/{1 0}/,B.comp/)
B,B.comp<=initN(init0(A,A.comp)))



4.8.3. XOR pipeline component network for LFSR

(XOR(A/{1 0}/,AB.comp/ B/{1 0}/,AB.comp/ =>
C/{1 0}/,C.comp/)

AB.comp<=?[C/{1<={[A/0 B/1] [A/1 B/0]}
0<={[A/0 B/0] [A/1 /B/1]} }
~C.comp])



4.8.4. The isolated LFSR network

The dependency expression

((=>) ((B C D E F O)/{1 0}/,*.comp)
F<=pipe1(E<=pipe0(D<=pipe1(C<=pipe0(B<=pipe0(pipe0(O
<=XOR(F XOR(XOR(B C) XOR(D E)))))))))

How to represent the two closures?

F is dependent through **pipe1** on **E** which is dependent through **pipe0** on **D** which is dependent through **pipe1** on **C** which is dependent through **pipe0** on **B** which is dependent through **pipe0** and through **pipe0** on **O** which is dependent through **XOR** on **F** and **XOR** which is dependent through **XOR** on **B** and **C**, and is dependent through **XOR** on **D** and **E**. The dependence of **O** on **F E D C** and **B** and the dependence of each of them on **O** closes each ring through the **XOR** network.

Localities **B**, **C**, **D** and **E** each associates to two places which flow cannot be represented with syntax relation but must be represented with name correspondence.

The inheritance expanded network expression.

((=>) (B/{1 0}/,B.comp C/{1 0}/,C.comp D/{1 0}/,D.comp
E/{1 0}/,E.comp F/{1 0}/,F.comp O/{1 0}/,O.comp)
F,F.comp<=pipe1(E,E.comp
<=pipe0(D,D.comp
<=pipe1(C,C.comp
<=pipe0(B,B.comp
<=pipe0(pipe0(O,O.comp
<=XOR(F,F.comp
XOR(XOR(B,B.comp C,C.comp)
XOR(D,D.comp E,E.comp)))))))))

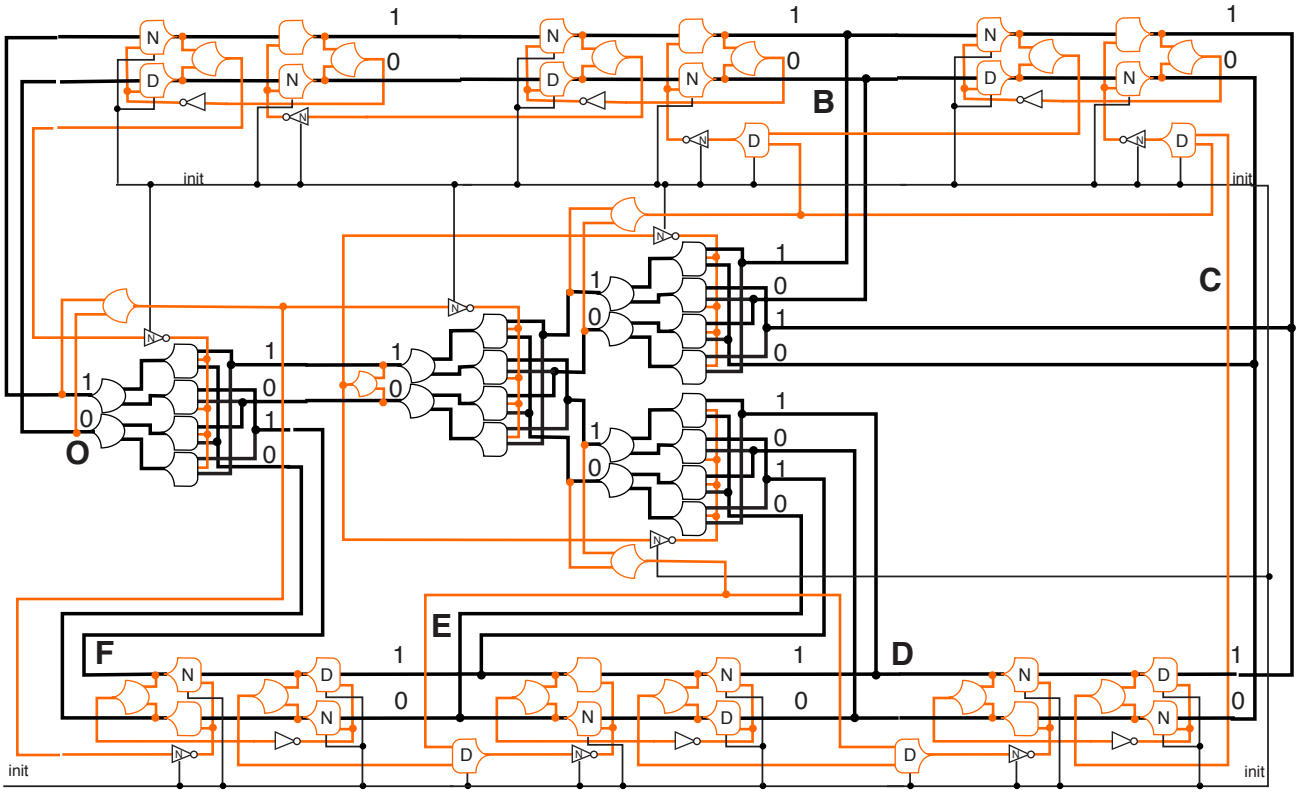


Figure 4.10. LFSR example network of multiple coupled rings

Figure 4.11 illustrates the **LFSR** network with 6 different wavefronts continually flowing around the 5 different rings of the network representing 6 differentnesses of time 5 of which interact through the shared **XOR** pipeline.

The 5 rings are color coded in Figure 4.11 to highlight their sharing of the peripheral oscillation networks.

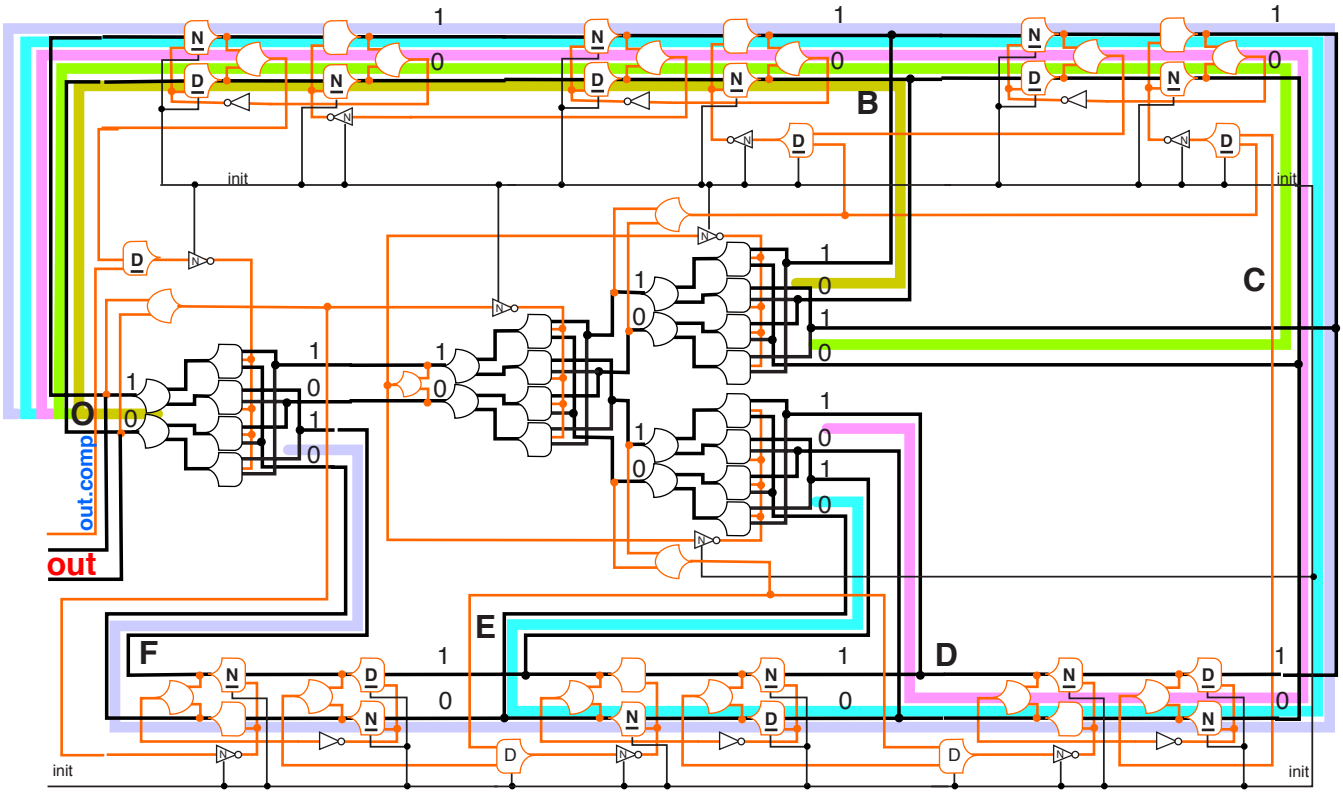


Figure 4.11. Coupled rings and their shared pipeline components.

4.8.5. The LFSR source network

Locality **O** of the **LFSR** can link to a pipeline network making it a source ring network. An assertion locality **out** dependent on locality **O** is added to the **LFSR** expression as a referenceable output.

The network expression.

```
(LFSR(  => out/{1 0}/,out.comp/)    ( B/{1 0}/,B.comp/  C/{1 0}/,C.comp/
      D/{1 0}/,D.comp/  E/{1 0}/,E.comp/  F/{1 0}/,F.comp/  O/{1 0}/,O.comp/  )
      F<=pipe1(E<=pipe0(D<=pipe1(C<=pipe0(B<=pipe0(pipe0(O
            <=[XOR(F XOR(XOR(B C) XOR.(D E))) ~out.comp])))))
            out<=O  )
```

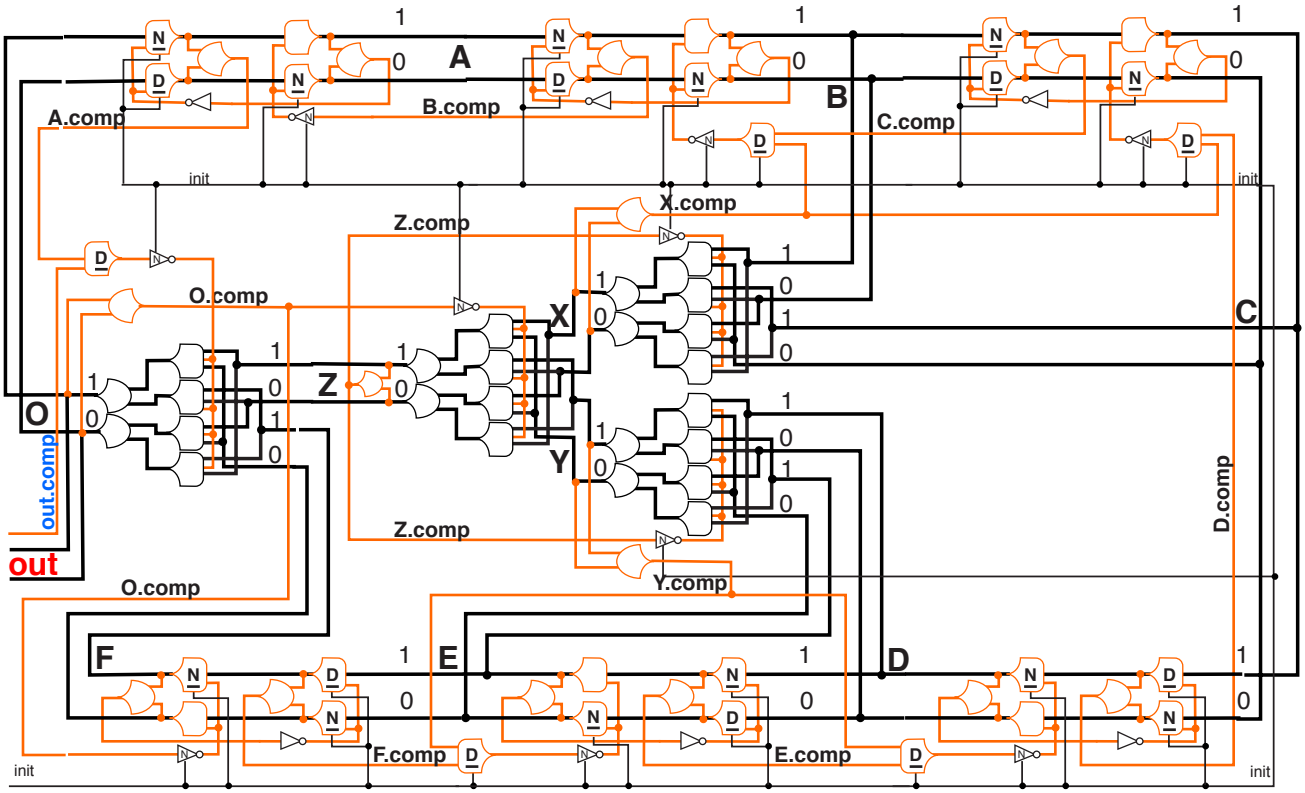



Figure 4.12. LFSR source network.

The LFSR is referenced as

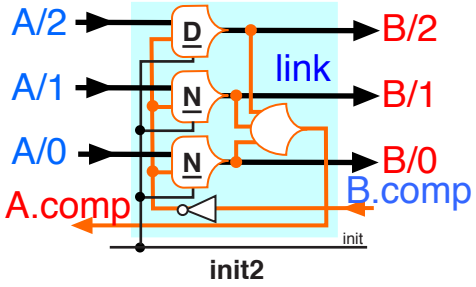
((LFSR( ))) or as dest<= LFSR( ) or as LFSR(=> dest)

4.9. The immersed ring network: engaging the environment

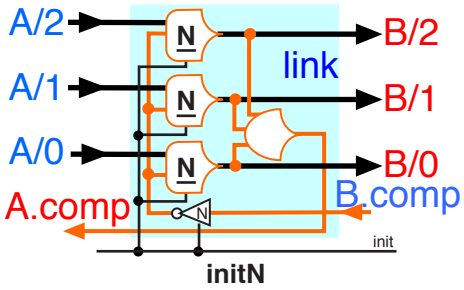
The immersed ring network of Figure 4.13 is a network of two coupled rings with an imposition portal to influence the environment and a sampling portal sensitive to the effects of the imposition on the environment forming an interaction network capable of remembering and learning.

4.9.1. The pipeline component networks for the immersed ring network

```
(init2(A/{2 1 0}/,A.comp/ =>
          B:2/{2 1 0}/,B.comp/)
  A.comp<=?B<=[A ~B.comp] )
```

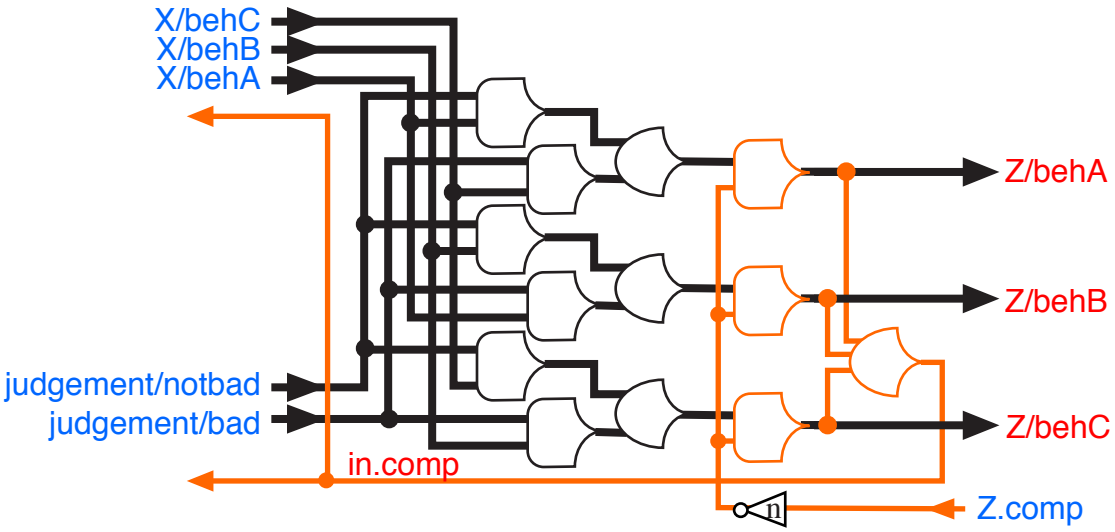



```
(initN(A/{2 1 0}/,A.comp/ =>
          B:N/{2 1 0}/, B.comp/)
  A.comp<=?B<=[A  ~B.comp]  )
```



4.9.2. The behavior interaction pipeline component network

```
(behmod(X/{behA  behB  behC}/,in.comp  judgement/{bad  notbad}/,in.comp =>
                                        Z/{behA  behB  behC}/,Z.comp)
in.comp<=?Z/{behA<=[{[X/behC  judgement/bad]
                     [X/behA  judgement/notbad]}  ~Z.comp]
            behB<=[{[X/behA  judgement/bad]
                     [X/behB  judgement/notbad]}  ~Z.comp]
            behC<=[{[X/behB  judgement/bad]
                     [X/behC  judgement/notbad]}  ~Z.comp]
          }  )
```



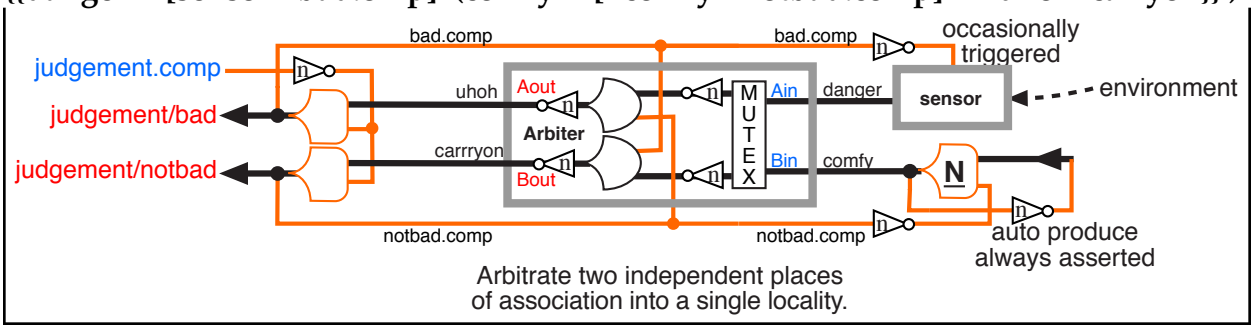
4.9.3. The arbitration pipeline component network

The arbiter (section 3.2.4)

{{Ain Bin => Aout Bout}}

arbitrates the continual liveness flow **comfy** with the occasional sensor flow **danger** into the network as a locality **judgement** of two mutually exclusive differentnesses.

(next(=> judgement/{bad notbad}/,judgement.comp/)
(carryon/ uhoh/ comfy/ danger/)
bad.comp<=judgement/bad<=[uhoh ~judgement.comp]
notbad.comp<=judgement/notbad<=[carryon ~judgement.comp]
{{danger<=[sensor ~bad.omp] (comfy<=[~comfy ~notbad.comp] => uhoh carryon}})



The dependency relations: **judgement** is dependent through the **arbiter** on **uhoh** which is dependent on the **sensor** and on **carryon** which is dependent on the **auto produce**. Locality **judgement** is enabled as a locality but each condition of the locality closes individually with its pre arbiter source.

4.9.4. The immersed ring network

The dependency expression.

((=>) (Z)
Z<=behmod(init2(initN(Z)) next( )))

The inheritance expanded expression.

((=>) (Z/{2 1 0}/,Z.comp)
Z, Z.comp <=behmod(init2(initN(Z, Z.comp)) next( )))

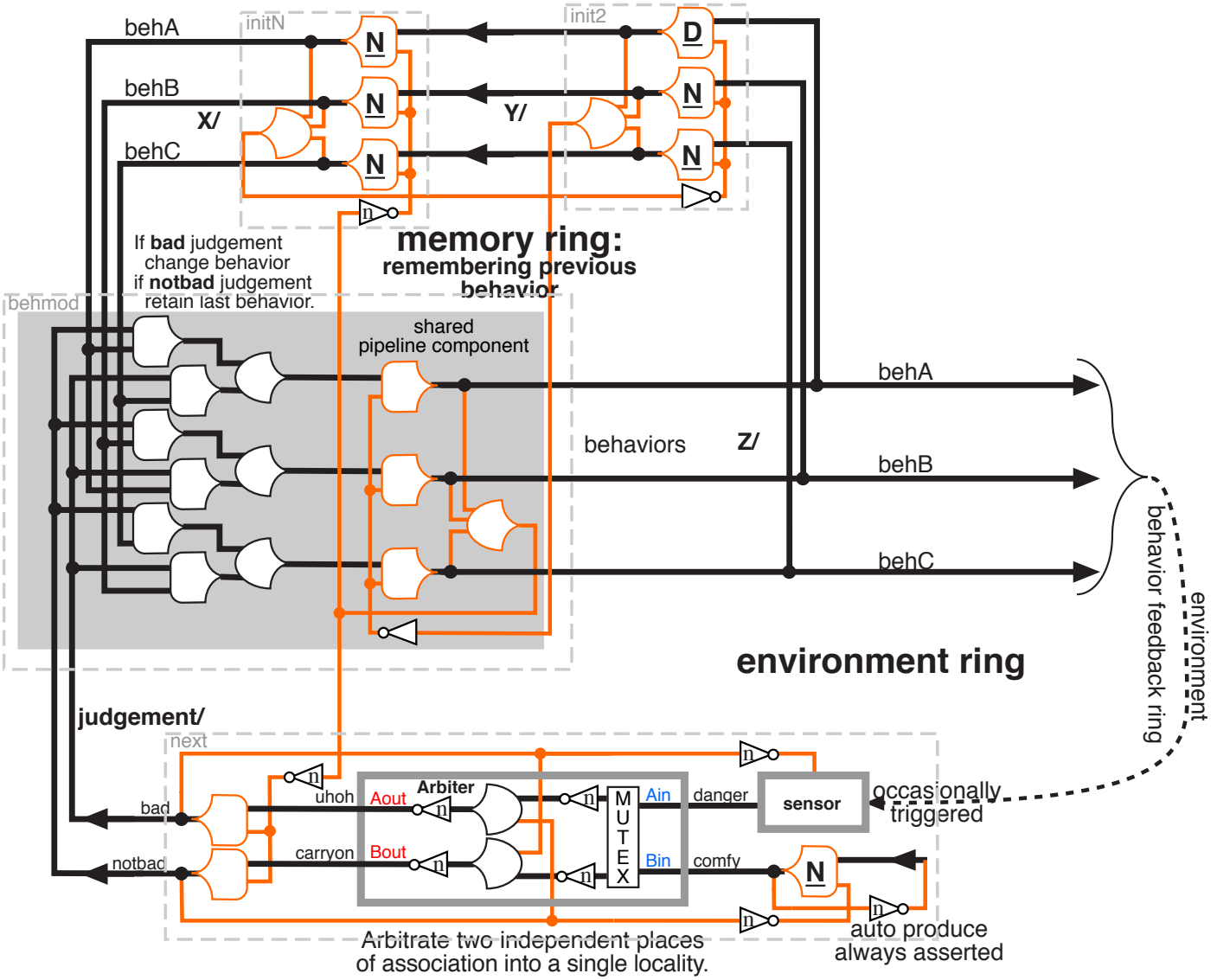


Figure 4.13. Linked rings with part of one ring flowing through the environment.

The upper ring of Figure 4.13 remembers the imposed behavior. The lower ring flowing partially through the environment determines whether the imposed behavior (section 3.12) should change or not. If the sensor's (section 3.12) closure is **D** the sensor will monitor the environment. If the monitoring exceeds some threshold the sensor will transition its output to **D**. When the closure becomes **N** it will transition its output to **N**. Upon which its closure will transition to **D** and the sensor will begin monitoring its input again. A **BAD** response from the sensor might be pain or frustration or danger. In the absence of a **BAD** response from the environment the sensor's output remains at **N** and the network remains alive from the auto produced, **NOTBAD** (section 3.11), continually presenting the same differentness to the environment.

The sensor **BAD** and the auto produce **NOTBAD** are happening independently and are not coordinated. They might even transition to **D** simultaneously. The arbiter receives the two uncoordinated flows **BAD** and **NOTBAD** and combines them into a single locality of mutually exclusive **judgement** conditions, presented to the network (section 3.2.4 and Appendix C) .The linked rings coupled through the interaction of **judgement** and **X** are continually alive continually imposing behavior on the environment at their intrinsic

throughput rate and only occasionally does the environment respond with an indication that behavior should be changed.

If **BAD** continues to be asserted the network will continue to change its behavior until it finds a behavior that no longer asserts **BAD**. When **BAD** ceases the auto produce will play through the arbiter with **NOTBAD** and the network will continue asserting the last behavior that did not elicit **BAD**. If none of the available behaviors unassert **BAD** then the network is in either mortal danger or chronic depression.

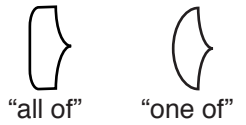
The immersed network through its spontaneously interacting differentnesses, is manipulating and appreciating the passive differentnesses of the external environment. While an immersed network can manipulate passive differentness, information, there is nothing manipulating the immersed network which is self contained and self determined. For example, while sequential computers are designed, implemented and manipulated by humans there is no designer and implementor of humans which must arise and behave entirely on their own merits.

4.10. The memory ring network: Stopping time

A wavefront presented to a closure link will wait indefinitely at the link for the closure enabling the wavefront to flow through the link. This behavior of indefinitely waiting for an enable can be used to store a wavefront and retrieve the wavefront. A ring can be configured such that a wavefront in the ring is caused to wait indefinitely at a closure link until an enable wavefront such as a read operation arrives. The memory ring has a binding portal through which write wavefronts and read write directives are presented. When the stored wavefront is enabled out of the pipeline ring it flows through the output of the pipeline ring and also flows around the ring back to the storage closure link to be read again. The memory ring can be written by enabling the wavefront at the wait closure link to be consumed making way for a new wavefront to flow into the pipeline ring to the closure link.

Wavefronts must be stopped and stored in **N**-**D** pairs so it requires a successive pair of closure links to stop a flowing wavefront. In Figure 4.14 the **D** completeness wavefront initialized to **0** is stored in locality **SD**. The following completely **N** wavefront is stored in locality **SN**.

The stretched behaviors:



represent behaviors spanning arbitrarily sized localities of differentness.

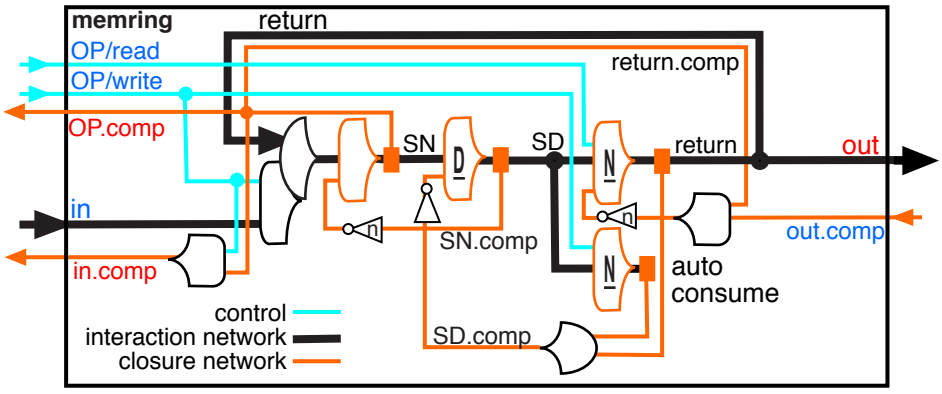


Figure 4.14. The memory ring network.

The memory ring network expression


```
(memring(in/[31-0]/{1 0}/,in.comp/  OP/{read write}/,OP.comp/  =>
                                    out/[31-0]/{1 0}/,out.comp/)
    (  return/[31-0]/{1 0}/,return.comp/  SN/[31-0]/{1 0}/,SN.comp/
       SD:0/[31-0]/{1 0}/,SD.comp/   )
SD.comp<={?return <=out<=[SD  OP/read  ~[out.comp  return.comp]]    /* read SD */
              ?[SD  OP/write] }                                      /* sink SD */
SN.comp<=?SD<=[SN  ~~SD.comp]                                        /* SN to SD */
in.comp<=[OP.comp<=return.comp<=?SN                                /* input or return to SN */
              <=[{return  [in  OP/write]}  ~SN.comp]  OP/write ] )
```

4.10.1. memring read behavior

When OP/read transitions to D and out.comp transitions to N the stored D completeness wavefront is enabled to locality (**return out**) leaving locality **SD** a D bubble. The completely N wavefront stored in locality **SN** flows into the D bubble of locality **SD** leaving locality **SN** a N bubble. The D completeness wavefront in **return** flows into the N bubble of locality **SN** and OP.comp transitions to D.

When OP.read transitions to N and out.comp transitions to D locality (**return out**) becomes a D bubble. The completely N wavefront in locality **SD** flows into locality (**return out**) leaving locality **SD** a N bubble. The D completeness wavefront in locality **SN** flows into locality **SD** leaving locality **SN** a D bubble. The completely N wavefront in locality **return** flows into locality **SN** and OP.comp transitions to N. out.comp transitions to N and the **memring** is ready for a next OP.

4.10.2. memring write behavior

When OP/write transitions to D the D completeness wavefront stored in locality **SD** is enabled to the auto consume locality leaving locality **SD** a D bubble. The completely N wavefront stored in locality **SN** flows into the D bubble of locality **SD** leaving locality **SN** a N bubble. locality in is enabled into locality **SN** and locality in becomes a D bubble. Locality OP.close transitions to D.

When OP/write transitions to N the auto consume locality becomes a D bubble. The completely N wavefront in **SD** is enabled to the auto consume locality leaving locality **SD** a N bubble. The D completeness wavefront in locality **SN** flows into locality **SD** leaving locality **SN** a D bubble. When locality **in** transitions to completely N the completely N wavefront flows into locality **SN** and OP.comp transitions to N and the **memring** is ready for a next OP.

OP/write and the **in** wavefront are **"all of"** related which must trace back to a common source, such as the decode of an instruction, that establishes the relation. The ring is initialized with a wavefront so there is always a wavefront stored in the memory. It is never empty. A read will always succeed and a write will always succeed.

4.10.3. A network of addressable memory rings: arranging time

Memory ring networks can be composed into a large addressable wavefront memory. Figure 4.15 expressed as a divergent-convergent network of **"one of"** related branches each branch containing a memory ring. Each memory ring represents differentness of both place of association and differentness of content. The wavefront stored in the ring is differentness of content. The address of the ring branch references differentness of place of association within the memory network which represents differentness of wavefront time. The memory network is indifferent to the differentness of wavefront content in any of its branch memory rings.

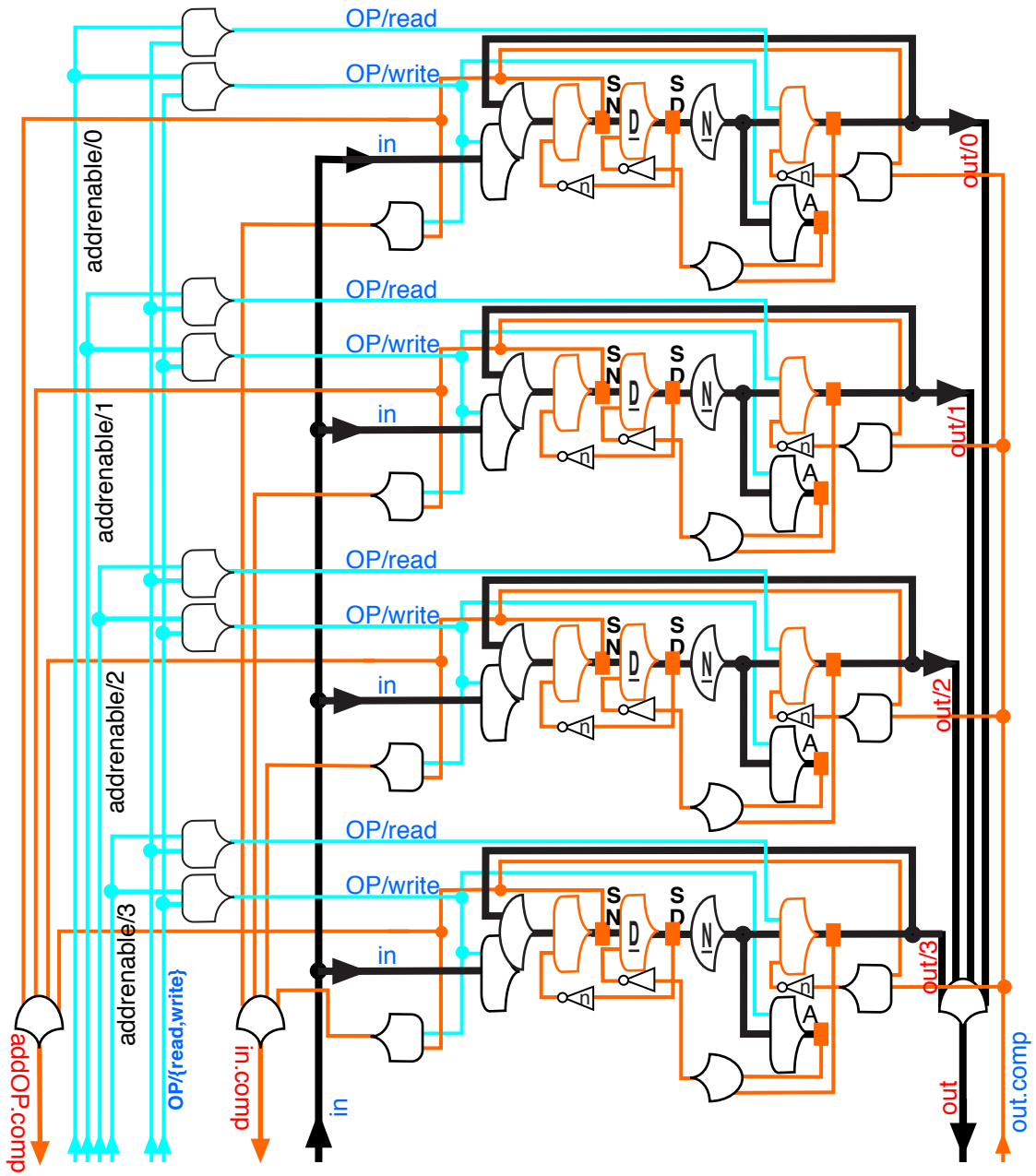


Figure 4.15. Addressable memory of memory rings.

Dependency expression for ram:

```
(ram(in  addrenable/{3 0}/  OP => out)
            (out/{3 0})
      readout<= {out/3-0<=memring(in  [addrenable/3-0  OP])}   )
```

addrenable/{3 0}/ is not inherited but is defined in ram specifying the size of the memory.
Inheritance expanded network expression.


```
(ram(in/[31-0]/{1 0}/,in.comp  addrenable/{3 0}/,addOP.comp
      OP/{read  write}/,addOP.comp => out/[31-0]/{1 0}/,out.comp)
      out,out.comp<= {memring(in,in.comp
                              [addrenable/0,addOP.comp OP,addOP.comp])
                      memring(in,in.comp
                              [addrenable/1,addOP.comp OP,addOP.comp])
                      memring(in,in.comp
                              [addrenable/2,addOP.comp OP,addOP.comp])
                      memring(in,in.comp
                              [addrenable/3,addOP.comp OP,addOP.comp])
                    }  )
```

memring being dependent on **addrenable/{3 0}** and on **out/{3 0}** is instantiated four times each receiving the corresponding components of **addrenable/x** and **out**. **in** and **OP** are distributed to all four intances of **memring**. **addrenable** determines which instance of **memring** receives **OP** and performs the specified **OP**.

Being able to stop a wavefront and to retrieve it on demand allows the rearrangement of wavefront flow through time and enables the possibility of controlling wavefront flow through dynamically constructed dependency networks with wavefront instances of interaction being stopped and saved in the addressable memory then being retrieved as needed. In particular the one behavior at a time sequential interpretation of dependency networks is enabled.

4.11. A most primitive sequence controller

Any dependency network expression can be decomposed into a sequence of simple behaviors that realize the dependency relations among the interaction network behaviors(section 1.1.1). The difficulty with one at a time sequencing of behaviors is that it forms a one dimensional behavior space in which wavefronts cannot flow directly from behavior to behavior but must flow indirectly through a means of delaying wavefronts and releasing them when needed in the sequence. The addressable memory of Section 4.10.3 fulfills this requirement enabling the sequential interpretation of dependency networks.

The other enabler of sequential interpretation is the sequence controller which realizes the one at a time sequence of behaviors by determining each next behavior, retrieving its input wavefronts from the memory flowing them through a specified interaction behavior and returning the result wavefront to memory.

4.11.1. Reducing a dependency network expression to a sequence of steps

An interpreter behavior is referenced as a full portal behavior reference (section 3.4.6.1).

behaviorname(localityA localityB => localityC)

An interaction network expression is reduced to an interpretable sequence by transforming dependency relations into full portal referenced interpreter behaviors and by mapping all localities into representations that fit into the wavefront memory. Finally, the behaviors needed to represent the three dimensional network in a one dimensional sequence of references are inserted into the sequence.

The expression from the LFSR example of section 4.8 serves as the reduction example.

```
(   (  =>  )    (B/{1 0}/  C/{1 0}/  D/{1 0}/  E/{1 0}/  F/{1 0}/  O/{1 0}/)
     F<=pipe1(E<=pipe0(D<=pipe1(C
           <=pipe0(B<=pipe0(pipe0(O
           <=XOR(F XOR(XOR(B C) XOR(D E)))))))))   )
```


An interaction network is expressed in terms of syntax relations and name correspondence relations. The first task is to re express all syntactic nesting relations as name correspondence associations among full portal behavior references. The nameless nested localities of the reference nesting relations are assigned names. The behavior references are related by name correspondence as full portal behavior references referencing the newly assigned names and are ordered to fulfill the dependency precedence relations of the interaction network expression.

4.11.1.1. The XOR full portal behavior references

XOR(D E => T3)
XOR(B C => T2)
XOR(T2 T3 -> T1)
XOR(F T1 => O)

XOR(D E => T3) and **XOR(B C => T2)** can be in any order but both must be before **XOR(T2 T3 -> T1)** which must be before **XOR(F T1 => O)**.

Some full portal references may need to be further reduced and substituted. Reduction proceeds until all references refer to behaviors realized by the sequence controller referencing localities that fit into the wavefront memory in an ordered sequence that fulfills the dependency relations of the original dependency network expression.

For the purpose of reduction only the interaction dependency relations are considered. The closure network is not regarded. The flow coordination of sequential interpretation is quite different from the closure flow coordination of the interaction network (section 4.11.2.2).

4.11.1.2. The pipex behavior references

The localities for the locality nested **pipe0** and **pipe1** behaviors are already named except for one which is assigned the name **T0**. The references are re expressed from locality nested references to full portal references in column **A** of Figure 4.16.

The **pipex** behaviors do double duty in the interaction network by initializing localities as well as moving wavefronts between localities. For the sequenced behaviors these duties are separated out in column **B** of Figure 4.16 with separate behaviors. Behavior **initx** initializes wavefronts in a locality and behavior **move** moves wavefronts from locality to locality.

A		B		C
pipex behaviors		**separation**		**integration**
				init0(T0)
		init0(T0)		**init0(B)**
		init0(B)		**init0(C)**
pipe1(E => F)		**init0(C)**		**init1(D)**
pipe0(D => E)		**init1(D)**		**init0(E)**
pipe1(C => D)		**init0(E)**		**init1(F)**
pipe0(B => C)	→	**init1(F)**	→	**XOR(D E => T3)**
pipe0(T0 => B)		**move(E => F)**		**XOR(B C => T2)**
pipe0(O => T0)		**move (D => E)**		**XOR(T2 T3 => T1)**
		move (C => D)		**XOR(F T1 => O)**
		move (B => C)		**move(E => F)**
		move (T0 => B)		**move(D => E)**
		move (O => T0)		**move(C => D)**
				move(B => C)
				move(T0 => B)
				move (O => T0)

Figure 4.16. Composing the sequence of dependent operations.

These full portal reference behaviors are placed in an order that preserves the dependency relations of the original interaction network expression. The **initx** behaviors can occur in any order among themselves but must all occur before any interaction behaviors in the ordered list of behaviors. In the **LFSR** network the first wavefronts flowing through the **XOR**s will be the initialized wavefronts not the first **move**d wavefronts. So the sequential order in column **C** of Figure 4.16 is to first perform the **initx**s and then perform the **XOR**s then perform the **move**s of the wavefronts. The **move**s of the wavefronts are ordered such that each locality is read and moved before the next behavior overwrites it.

4.11.1.3. One dimensional complexity

The interaction behaviors of the above list in column **C** of Figure 4.16 (excluding the init behaviors) remain a valid dependency network expression. The behaviors can be shuffled into any order and the name correspondence relations still construct the dependency network minus the inits. For sequential interpretation, however, there are only a few specific orderings of behaviors that will work and there are additional behaviors not derived from the dependency network expression that have to be added for sequential interpretation to work.

An ordered list of behaviors to be interpreted one behavior at a time in sequence forms a one dimensional behavior space that cannot, on its own, characterize the connectivity of a three dimensional interaction network. Because of concurrency and conditionality the next behavior to be realized cannot always be the next behavior in the one dimensional sequence. It is necessary to conditionally and unconditionally jump around in the one dimensional sequence of behaviors tunneling through the one dimensional space. The sequence controller must support the additional behaviors that enable conditional and unconditional jumps within the one dimensional sequence.

Each directive is assigned a unique name so it can be referenced. A name range with an inherent order and easily determined nextness such as numeric is most convenient. In column **A** of Figure 4.17 the assigned names are numeric and an unconditional **jump** is inserted as the last directive closing the ring of dependency. The interpretation begins with the directive named **100**, progresses to directive named **116** then continues with the directive named **106** forming an infinite loop of interpretation emulating the continual behavior and wavefront flow of the **LFSR** ring network.

	A		B		C
100	**init0(T0)**	**100**	**init0(T0)**	**100**	**2(20)**
101	**init0(B)**	**101**	**init0(B)**	**101**	**2(11)**
102	**init0(C)**	**102**	**init0(C)**	**102**	**2(12)**
103	**init1(D)**	**103**	**init1(D)**	**103**	**3(13)**
104	**init0(E)**	**104**	**init0(E)**	**104**	**2(14)**
105	**init1(F)**	**105**	**init1(F)**	**105**	**3(15)**
106	**XOR(D E => T3)**	**106**	**XOR(D E => T3)**	**106**	**4(13 14 => 23)**
107	**XOR(B C => T2)**	**107**	**XOR(B C => T2)**	**107**	**4(11 12 => 22)**
108	**XOR(T2 T3 -> T1)**	**108**	**XOR(T2 T3 => T1)**	**108**	**4(22 23 => 21)**
109	**XOR(F T1 => O)**	**109**	**XOR(F T1 => O)**	**109**	**4(15 21 => 16)**
112	**move(E => F)**	**110**	**wait**	**110**	**6**
113	**move(D => E)**	**111**	**move(O => out)**	**111**	**5(16 => 24)**
114	**move(C => D)**	**112**	**move(E => F)**	**112**	**5(14 => 15)**
115	**move(B => C)**	**113**	**move(D => E)**	**113**	**5(13 => 14)**
116	**move(T0 => B)**	**114**	**move(C => D)**	**114**	**5(12 => 13)**
117	**move(O => T0)**	**115**	**move(B => C)**	**115**	**5(11 => 12)**
116	**jump (106)**	**116**	**move(T0 => B)**	**116**	**5(20 => 11)**
		117	**move(O => T0)**	**117**	**5(16 => 20)**
		118	**jump(106)**	**118**	**7(106)**

Figure 4.17. Forming and encoding the one dimensional sequence of behaviors.

The **LFSR** source network of Section 4.8.5 can be expressed by inserting a simple interface protocol. In column **B** of Figure 4.17 a **wait** directive is inserted after the last **XOR** directive and a **move** directive is inserted to move the wavefront in locality **O** to locality **out**.

Interpretation begins at **100** and halts at the **wait** directive with the **XOR**s having delivered their output wavefront to locality **O**. An external reference requests (closes with) the wavefront of locality **O**. The wait falls through and the next behavior moves the wavefront in **O** to **out**. The sequence of behaviors continues with the **move** behaviors moving the wavefronts presented to the **XOR**s from locality to locality. The wavefronts then flow through the **XOR**s with their output wavefront flowing to locality **O**. The interpretation then **wait**s for the next request.

4.11.1.4. Mapping references to memory addresses and to behavior codes

The behavior names are translated into a code that the interpreter can understand to select the appropriate behavior. Each locality name is assigned a numeric memory address name.

numeric locality names		behavior codes
B -> 11	T0 -> 20	init0 -> 2
C -> 12	T1 -> 21	init1 -> 3
D -> 13	T2 -> 22	XOR -> 4
E -> 14	T3 -> 23	move -> 5
F -> 15	out -> 24	wait -> 6
O -> 16		jump -> 7

The ordered list of behaviors with the reassigned names is in column **C** of Figure 4.17. The behavior code and the memory address names of each behavior are contrived to fit into a single unit of wavefront memory that is referenced with one memory address forming an interpretation behavior that a sequence controller can read and understand specifying a flow from wavefront memory through the referenced behavior and back to wavefront memory. The entire expression is now in terms of a sequence of full portal references to numerically coded behaviors realized by the interpreter referencing numeric memory addresses.

The incoherent transition behavior of a self determined interaction network becomes temporally and spatially regimented behavior in its sequentially interpreted counterpart.

4.11.2. The sequence controller

The sequence controller of Figure 4.18 is a network of coupled and linked rings that reads each behavior from memory and manages the flow of wavefronts from an addressable memory through specified interaction behaviors whose results flow back to the memory. There is the **program counter ring** network which determines from the current address and from the the type of the previous behavior, branch or not branch, the next address from which to read a behavior. There is the **decoder ring** network which reads and decodes the behavior dispatching the behavior code and the memory addresses to the **ALU/memory ring** network. The **ALU/memory ring** network accesses the input wavefronts from memory, flows them through the specified behavior and places the result wavefront back in memory. There is also the **branch ring** coupled through the **ALU** and the **program counter** which determines the conditional of a branch.

The sequence of behaviors for the interaction network being interpreted are placed in the memory in numeric order. The initial wavefronts of localities are placed in their memory locations. The program counter initializes the next behavior address to the first behavior to be read, in this case **100**. The address is passed to the decoder and the behavior is read, decoded and realized. The sequence controller then cycles through the behaviors. The resulting sequence of behaviors realizes the source LFSR network constructing the network's behavior extended through time and memory.

4.11.2.1. The exposed binding portal

The sequential interpreter is not a completely self determined network that can explore its environment entirely on its own. It has an exposed binding portal in that any program can be loaded into its memory from an external environment. Once a program is loaded, however, it self determinately realizes the program.

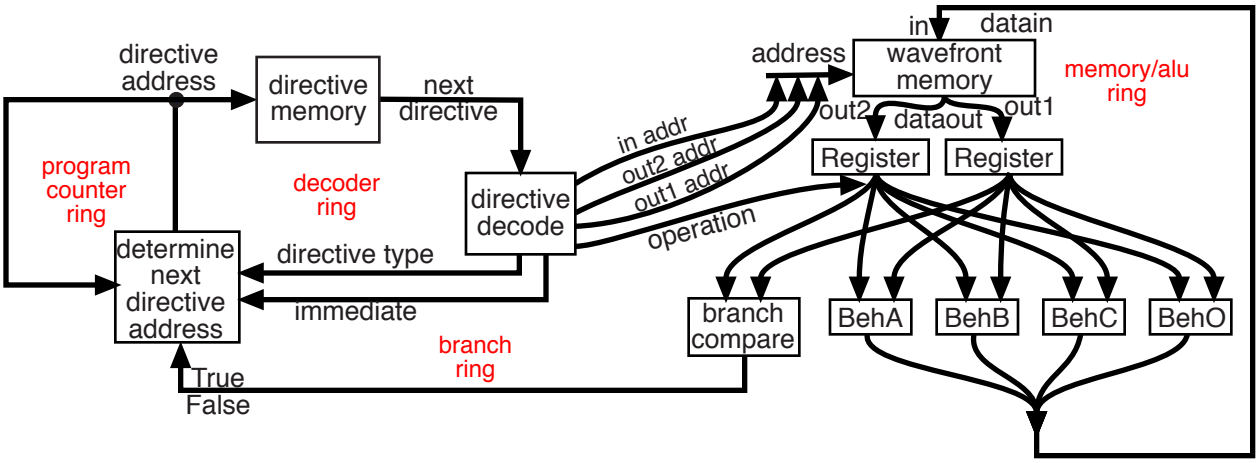


Figure 4.18. The sequential interpreter and its linked rings.

A sequential controller however might be outfitted with imposition and sensor behaviors and loaded with a never ending program that is essentially self determined and capable of independently exploring its environment, a robot.

4.11.2.2. Sequential flow coordination

Sequential interpretation requires that each directive is completed with its asserted output written to memory before the next sequential behavior is begun. Completeness of output ensures the stable completeness of input presentation for subsequent behaviors. This is different from the dependency network for which the completeness of output of a behavior ensures the completeness of its own input. In both cases interaction behavior is coordinated in terms of completeness of output assertion progressively accumulating to ensure completeness of input presentation.

4.11.2.3. Scale, generality and universality but not fundamentality

The sequence controller and addressable memory form a *universal sequential interpreter*, an dependency network capable of interpretively realizing the behavior of any other dependency network. Realizing every dependency network as a unique physical network is not practical and does not scale. But being able to express any dependency network directly as a dependency network, translate it to a sequential expression and then to realize the dependency network expression dynamically constructed through time and memory with a simple sequential interpreter that can be indefinitely replicated does scale.

The dynamic and conditionally varying construction of dependency relations through time and memory provides a flexibility and generality of realizability not possible with statically realized networks.

The interpreter itself is a computation that can realize any other computation but a universal interpreter can take many disparate forms and does not in itself embody any fundamental essence of computation.

4.11.2.4. Expressive generality

Why express one of the multitudes of possible sequences when you can express the one unique dependency expression that all the sequences must derive from and which you must understand to express any one of the sequences and have it mechanically translated, compiled, to an appropriate sequential expression (section 1.1.1).

4.11.2.5. Again, nothing new or extrinsic

No new primitivity or extrinsic capability has been introduced. The addressable memory and sequence controller are realized entirely in terms of the primitive behaviors of section 3.2.[1] The sufficiently expressive primitive behaviors have finally bootstrapped sequential interpretation.

The derivation rationale of the sequential interpreter did not appeal to a mathematician with pencil and paper nor to the notion of algorithm. There was no appeal to Boolean logic, to timing analysis or to a timing referent such as a clock. Sequential one at a time interpretive stepping was not assumed a priori but emerged as a possibility of universal realizability with the emergence of the addressable memory in which wavefronts could be stored and retrieved.

4.12. A quest fulfilled

Journeying from the pure condition differentiation expression of Chapter 2 through the pure association differentiation expressions of Chapter 3 to the temporal differentiation expressions of Chapter 4 computational interaction with all concurrent relations is fully accounted in terms of *differentness spontaneously and dependently interacting and changing* with primitive interaction behaviors (section 3.2) that *reliably express primitive concurrency behavior* and that *realize a most primitive sequential interpreter* bootstrapping sequential interpretation.

4.12.1. Counter example

Sequence control and Boolean logic are considered fundamental to computation. Dependency expression presents a counter example expressing deterministic complex computation containing no Boolean logic expression, no sequence control or any other form of explicit control, possessing no stable samplable state and possessing no common behavioral referent such as a clock or a mathematician. Furthermore, sequence control and Boolean logic are derivable from dependency expression.

4.12.2. Full circle

A self determined interaction network arises able to replicate itself and evolve into a complex self determined interaction network that invents differentness interaction mapping with an exposed binding portal, that becomes a controlling external environment using interaction as a tool to ask questions and receive answers and that becomes fascinated with the limitations of the exposed binding portal.

4.12.3. Still not done

There remains one final stage of the journey, a walk through the spectrum of differentiation relating the dimensions of condition differentiation and association differentiation.

1. The author in collaboration with Igor Ilyukhin has implemented a RISC V 32IM as a structure of coupled rings in terms of the primitive interaction behaviors of section 3.2.

Chapter 5:
The spectrum of Differentiation

Any interaction of differentness is represented as a coextensive collaboration between condition differentness and association differentness. The spectrum of differentiation, Figure 5.1, embodies this collaboration with pure condition differentiation at one end of the spectrum and pure association differentiation at the other end with various proportions of collaboration in between.

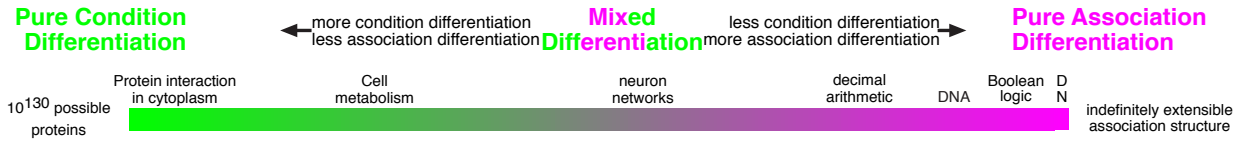


Figure 5.1. The spectrum of differentiation.

5.1. The collaboration

Different conditions are different within a same place of association. A same condition is different within different places of association. A range of mutually exclusively different conditions extends the expression of differentness of a same place of association. Different places of association asserting the same range of different conditions extends the expression of differentness of the same range of different conditions.

With decimal place value numbers, for instance there are 10 different numeral conditions **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8** and **9**. One differentness of place of association in a decimal place value number can represent 10 different numeral conditions extending the range of differentness expression of the one place by ten differentnesses. Four different places of association asserting the same numeral condition, 3333, extends the range of differentness expression of the one numeral condition by four differentnesses. Each numeral condition is different by virtue of its differentness of place. Condition differentness and association differentness collaborate to express a large range of mutually exclusive differentness.

With binary place value numbers each differentness of place of association is extended to two differentnesses **0** and **1**. Each place of association of a binary number can express only **0** or **1** so it takes more places of association to express a comparable decimal number. Decimal number **1023** requires four decimal places while the comparable binary number **1111111111** requires 10 binary places. Two representations of a same differentness with different proportions of condition differentiation and association differentiation.

Twenty six letter conditions asserted one at a time at different places of association relations form words. Words associate to form sentences. Sentences association to form paragraphs and so on. Japanese uses an enormous range of different symbol conditions asserted at different places of association to form its words and sentences. Again different proportions of condition differentness and association differentness to express comparable meaning differentness.

The spectrum of differentiation encompasses this proportionality of collaboration of mutual extension between condition differentness and association differentness in the representation of mutually exclusive differentnesses and their interactions providing a unifying foundation of expressivity among what have been considered to be quite disparate forms of representing differentness and the interaction of differentness.

5.2. A walk along the spectrum

The collaboration is illustrated by walking one temporal instance of one specific interaction of differentnesses along the spectrum with condition differentiation and association differentiation collaborating in various proportion to represent the one specific interaction.

One differentness from the group of three mutually exclusive differentnesses named **U**, **V**, **W** and one differentness from the group of three mutually exclusive differentnesses named **X**, **Y**, **Z** associate and interact producing one of nine mutually exclusive differentnesses named **A**, **B**, **C**, **D**, **E**, **F**, **J**, **K**, **L**. The example interaction is shown in Figure 5.2 as an interaction mapping table and as a list of interaction dependency relations.

mapping table

	U	V	W
X	A	B	C
Y	D	E	F
Z	J	K	L

specifies interaction→ dependencies

[W Z] => L
[V Z] => K
[U Z] => J
[W Y] => F
[V Y] => E
[U Y] => D
[W X] => C
[V X] => B
[U X] => A

Figure 5.2. Example interaction.

The tradeoffs between the proportion of condition differentiation and association differentiation are considered in terms of the example interaction at various locations along the spectrum characterized in terms of the amount of condition differentiation available. The walk begins at the pure condition end of the spectrum with sufficient condition differentiation to express the example interaction at a single place of common association and continues by constraining condition differentiation until there is only one available differentness condition at the pure association end of the spectrum.

5.3. With fifteen available differentness conditions

With fifteen available differentness conditions named **A**, **B**, **C**, **D**, **E**, **F**, **J**, **K**, **L**, **U**, **V**, **W**, **X**, **Y**, **Z** the example interaction can be represented with pure condition differentiation (Chapter 2 for a more complex example of interaction expressed purely in terms of condition differentiation) at a single place of common association as illustrated with the bag in Figure 5.3. The bag begins empty. One of conditions **X**, **Y**, **Z** and one of conditions **U**, **V**, **W** enter the bag and interact with the interaction propensities expressed in the mapping table producing one of conditions **A**, **B**, **C**, **D**, **E**, **F**, **J**, **K**, **L**. The result condition exits the bag leaving the bag empty bounding one temporal instance of one specific interaction of differentness. The interaction mapping is the same as Figure 5.2.

	U	V	W
X	A	B	C
Y	D	E	F
Z	J	K	L

[W,Z] -> L
[V,Z] -> K
[U,Z] -> J
[W,Y] -> F
[V,Y] -> E
[U,Y] -> D
[W,X] -> C
[V,X] -> B
[U,X] -> A

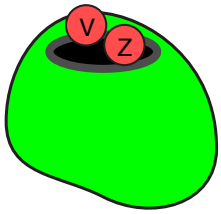


Figure 5.3. Fifteen differentness conditions, 1 shaking bag.

With only one place of common association and with associations inside the bag being indeterminate there is no differentiation in terms of association. The differentnesses and their interaction is expressed purely in terms of condition differentnesses and their interaction propensities mapping input differentness conditions to output differentness conditions.

The interaction dependency expression as relations of names of conditions:

{L<=[W Z]
K<=[V Z]
J<=[U Z]
F<=[W Y]
E<=[V Y]
D<=[U Y]
C<=[W X]
B<=[V X]
A<=[U X] }

There is no referencing or binding in a pure value expression. Conditions are sufficiently differentiating. They just show up and interact.

All possible interaction mappings can be expressed by specifying the interaction propensities in the mapping table. For instance:

mapping table

	U	V	W
X	B	A	D
Y	D	B	A
Z	C	E	B

specifies interaction→ dependencies

W Z => B
V Z => E
U Z => C
W Y => A
V Y => B
U Y => D
W X => D
V X => A
U X => B

The interaction is specified entirely in terms of condition differentiation.

5.4. Constrained to nine available differentness conditions

Nine conditions are sufficient to represent the output differentnesses but six of the conditions must also be used to represent the input differentnesses.

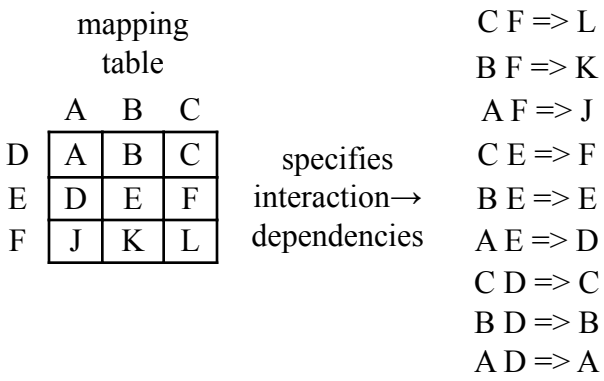


Figure 5.4. Interaction mapping with nine available differentness conditions.

Input differentness conditions must be differentiated from the identical output differentness conditions. This is represented with an association interaction behavior that isolates input conditions from identical output conditions establishing isolated localities of differentness for its inputs and its output, see Figure 3.4 and section 3.1.3.

An association interaction behavior, as explained in Section 3.1.6, continually receiving input interaction conditions and continually asserting an output interaction condition requires a condition named **N** explicitly representing absence of interaction condition as well as the monotonic transitioning between interaction condition completeness and completely **N** including the ability of the interaction association behavior to recognize transitions of input to interaction condition completeness and to completely **N**. The condition named **N** is included as integral to this discussion and will be explicitly referenced in the interaction behavior mapping tables and in the interaction dependency expressions.

Interaction no longer occurs in terms of freely associating conditions but now occurs inside the association interaction behavior in terms of directly associated conditions. The interaction is characterized by the mapping table on the right of Figure 5.5 which includes the nine differentness conditions plus **N** with the **–** indicating no transition. The interaction mapping behavior transitions its asserted output to an interaction differentness condition only when both presented inputs are interaction differentness conditions (interaction differentness completeness) and transitioning its asserted output condition to **N** only when both presented inputs are differentness condition **N** (completely **N**) representing emptiness of interaction differentnesses (Section 3.1.6). When the presented input is not complete, i.e. there is one **N** and one interaction differentness condition presented, the interaction behavior does not transition its asserted output condition (section 3.2).

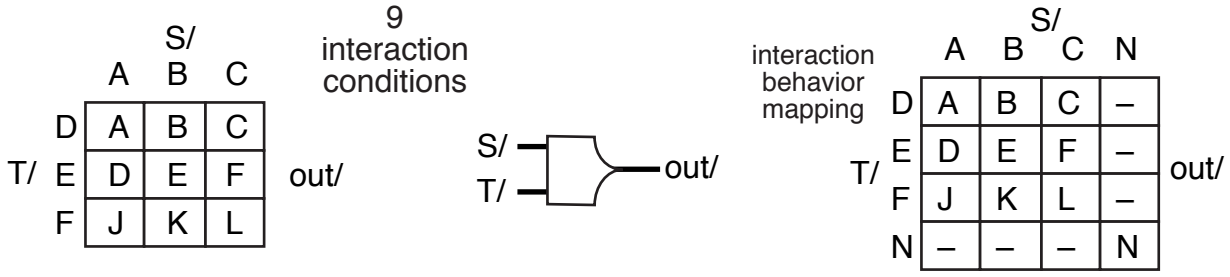


Figure 5.5. Nine differentness conditions with one association interaction behavior.

The interaction behavior forms a simplest of interaction networks. Each differentness now is expressed as a differentness of place of association combined with a differentness of condition. Locality **S** can assert three interaction differentnesses **S/C**, **S/B**, **S/A** and **S/N**. Locality **T** can assert three interaction differentnesses **T/F**, **T/E**, **T/D** and **T/N**. Locality **out** can assert differentnesses nine interaction differentnesses **out/L**, **out/K**, **out/J**, **out/F**, **out/E**, **out/D**, **out/C**, **out/B**, **out/A** and **out/N**. Remember that **N** is not an interaction differentness. The differentnesses asserted with identical conditions at different places of association are different by virtue of the differentness of place of association. **T/F** and **out/F** are different by virtue of the associational differentness of **T** and **out**.

An interaction begins empty with **S/N T/N** and **out/N.** Inputs **S/** and **T/** transition to interaction conditions and **out/** transitions to the determined interaction condition. **S/** and **T/** transition back to **N**, and **out/** transitions to **N** leaving the interaction behavior empty bounding one temporal instance of one specific interaction of differentness.

The interaction dependency expression as relations of names of conditions and names of places of association:

(9condexamp(S/{C B A N} T/{F E D N} => out/{L K J F E D C B A N})
out/{L<=[S/C T/F]
K<=[S/B T/F]
J<=[S/A T/F]
F<=[S/C T/E]
E<=[S/B T/E]
D<=[S/A T/E]
C<=[S/C T/D]
B<=[S/B T/D]
A<=[S/A T/D]
N<=[S/N T/N] })

All possible interaction mappings can be expressed by arranging the output differentnesses in the mapping table. The burden of representing differentiation and interaction is now shared between condition differentiation and association differentiation.

5.5. Constrained to six available differentness conditions

Six differentness conditions named **A**, **B**, **C**, **D**, **E**, **F** are still sufficient to represent the input differentnesses but six differentness conditions are not sufficient to represent the nine output conditions so the output differentness **out** must now be represented as two associated output conditions asserted by two association interaction behaviors asserting locality **out** with two component localities **Y** and **Z**. Input locality **S/** can assert one of the interaction differentness conditions **A** or **B** or **C**. Input locality **T/** can assert one of the interaction differentness conditions **D** or **E** or **F**. The example interaction mapping is shown in Figure 5.6.

The specific interaction mapping.

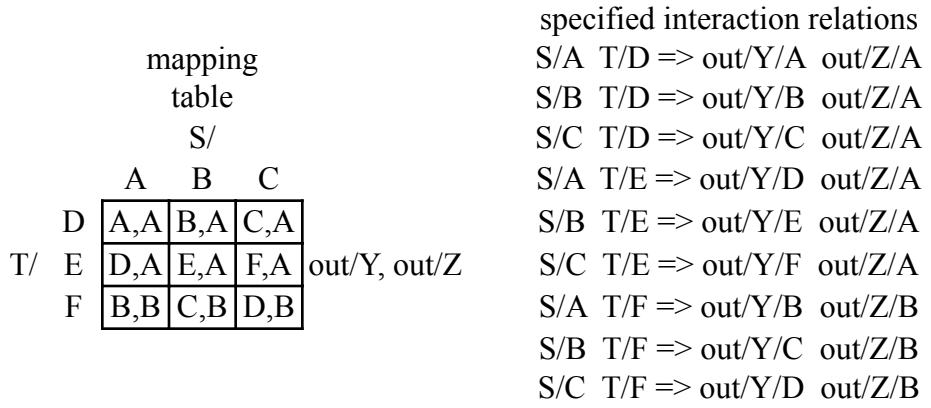


Figure 5.6. Interaction mapping with six available differentness conditions.

The interaction in Figure 5.7 is now expressed as a network of two association interaction behaviors asserting the two output conditions. The input conditions asserted by **S** and **T** are fanned out to the two interaction behaviors. The interaction mapping **6cond1** asserts **out/Y** and the interaction mapping **6cond2** asserts **out/Z**. The availability of all possible interaction mappings means that any arbitrary mapping to the output can be represented.

An interaction begins empty with **S/N T/N** and **out/Y/N** and **out/Z/N**. inputs **S/** and **T/** transition to interaction differentness conditions and **out/Y** and **out/Z** transition to the determined interaction differentness conditions. **S/** and **T/** transition back to **N**, and **out/Y** and **out/Z** transition to **N** leaving the interaction network empty bounding one temporal instance of one specific interaction of differentness.

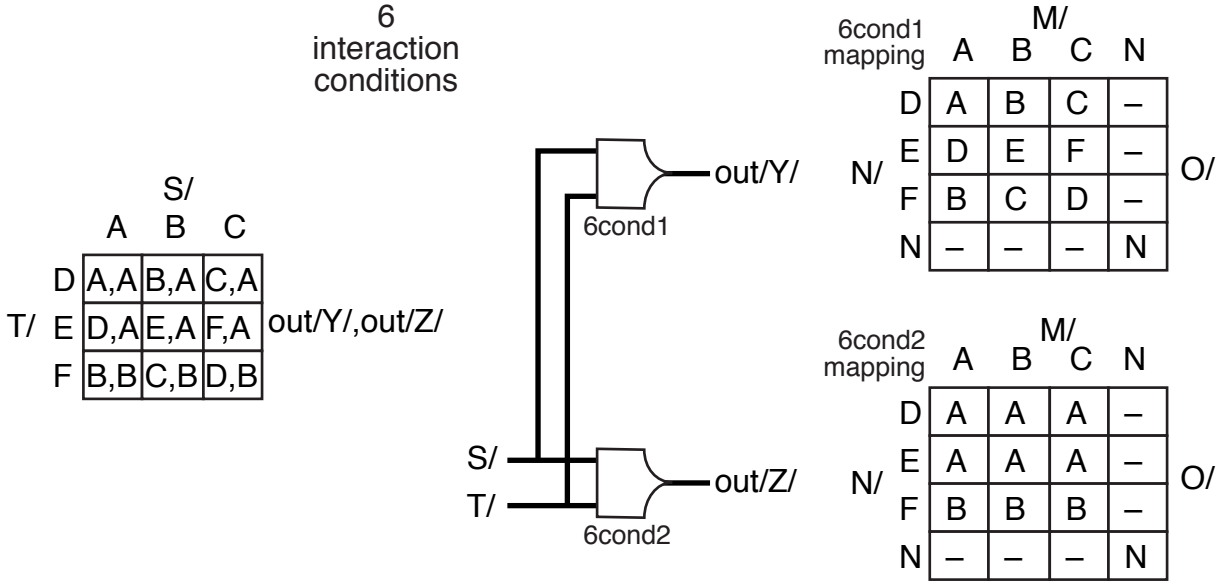


Figure 5.7. Six differentness conditions with two association interaction behaviors.

The dependency expressions for the component association interaction behaviors and for the network as relations of names of conditions and names of places of association:


```
(6cond1(M/{C B A N}  N/{F E D N}  => O/{F E D C B A N})
                O/{A<=[M/A  N/A]
                   B<={[M/B  N/D]  [M/A  N/F] }
                   C<={[M/C  N/D]  [M/B N/F] }
                   D<={[M/A  N/E]  [M/C N/F] }
                   E<=[M/B  N/E]
                   F<=[M/C  N/E]
                   N<=[M/N  N/N]  } )

(6cond2(M/{C B A N}  N/{F E D N}  => O/{F E D C B A N})
                O/{A<={[M/C  N/D]  [M/B  N/D]  [M/A  N/D]
                       [M/C  N/E]  [M/B  N/E]  [M/A  N/E] }
                   B<={[M/C  N/F]  [M/B  N/F]  [M/A  N/F]}
                   N<=[M/N  N/N]  } )

(6condexamp(S/{C B A N}  T/{F E D N}=> out/[Z  Y]/{F E D C B A N})
                6cond1(S  T => out/Y)
                6cond2(S  T => out/Z)    )
```

Expressions **6cond1** and **6cond2** express the mappings of the interaction behaviors. The expression **6condexamp** referencing **6cond1** and **6cond2** expresses how the mappings dependently associate as a network to realize the interaction of **S/** and **T/** to **out/[Y Z]/**. The network as a whole becomes the element of interaction that is singularly referencable from interaction to interaction

All possible interaction mappings can be expressed by arranging the output differentnesses in the mapping tables of the interaction behaviors to represent any desired mapping of combined output conditions.

There is less condition differentiation and more association differentiation.

5.6. Constrained to three available differentness conditions

There are two ways to constrain the expression of condition differentiation. First, the number of available conditions can be limited with the availability of association interaction behaviors capable of representing all the possible interaction mappings. Second, the available interaction mappings can also be limited. These two constraints can be conveniently illustrated and discussed in the context of three available interaction differentness conditions

5.6.1. First: all possible interaction mappings available

Three interaction differentness conditions named **A**, **B**, **C** are still sufficient to represent each input differentnesses. The nine output differentnesses can still be represented with two association interaction behaviors asserting locality **out** as two component localities **Y** and **Z** which must now be associatively ordered to differentiate **out/Y/A out/Z/B** from **out/Y/B outZ/A**. With six interaction differentness conditions the **Y** and **Z** outputs did not necessarily have to be ordered to distinguish the nine output differentnesses. Input locality **S/** asserts one of the interaction differentness conditions **A** or **B** or **C**. Input locality **T/** asserts one of the interaction differentness conditions **A** or **B** or **C**. The example interaction mapping is shown in Figure 5.8.

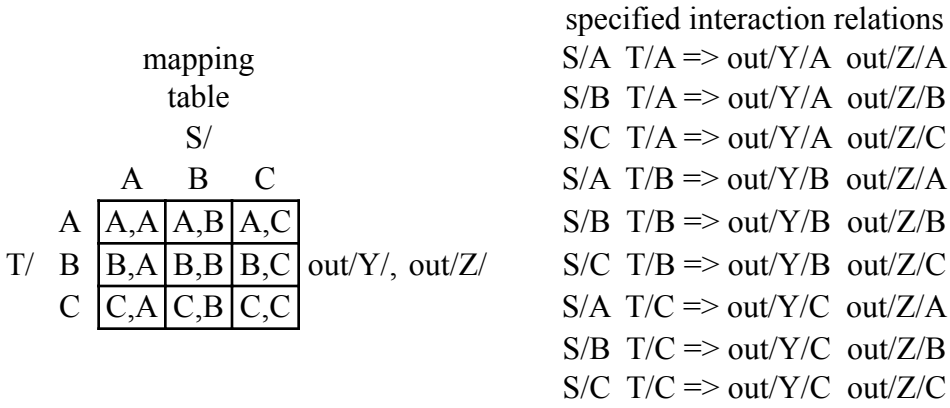


Figure 5.8. Interaction mapping with three available differentness conditions.

The interaction in Figure 5.9 is still expressed as a network of two association interaction behaviors asserting the two output conditions. The input conditions asserted by **S** and **T** are fanned out to the two interaction behaviors. The interaction mapping **3cond1** asserts **out/Y** and the interaction mapping **3cond2** asserts **out/Z**. The availability of all possible interaction mappings means that any arbitrary mapping to the output can be represented.

An interaction begins empty with **S/N T/N** and **out/Y/N** and **out/Z/N.** inputs **S/** and **T/** transition to interaction differentness conditions and **out/Y/** and **out/Z/** transition to the determined interaction differentness conditions. **S/** and **T/** transition back to **N**, and **out/Y/** and **out/Z/** transition to **N** leaving the interaction network empty bounding one temporal instance of one specific interaction of differentness.

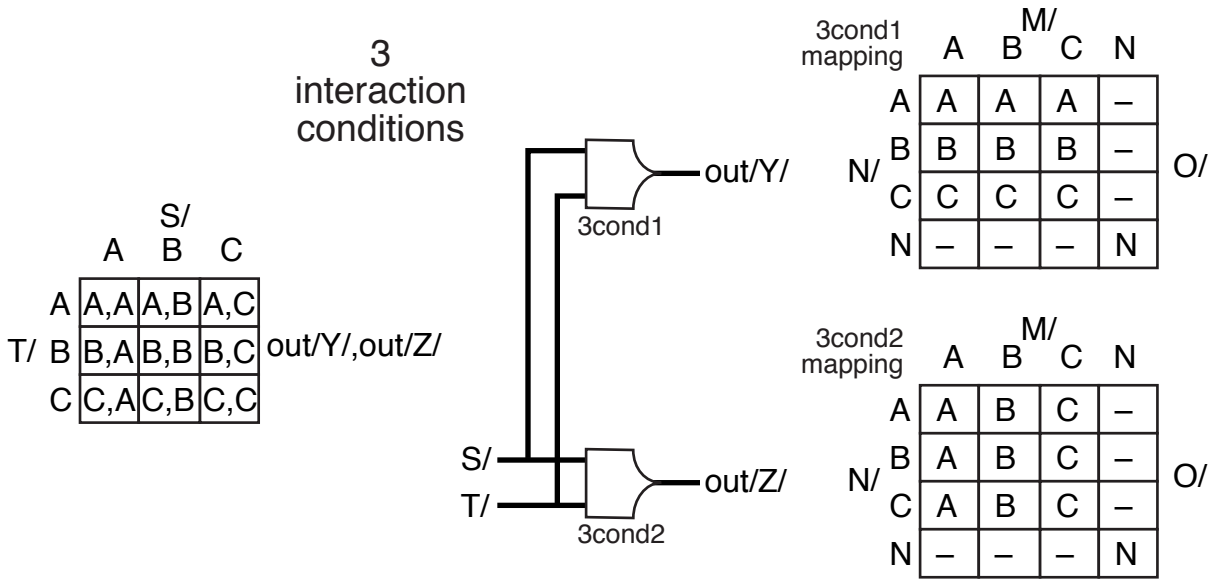


Figure 5.9. Three differentness conditions with two association interaction behaviors.

The dependency expressions for the component association interaction behaviors and for the network as relations of names of conditions and names of places of association.

(3cond1(M/{C B A N} N/{C B A N} => O/{C B A N})
O/{A<={[M/A N/A] [M/A N/B] [M/A N/C]}
B<={[M/B N/A] [M/B N/B] [M/B N/C]}
C<={[M/C N/A] [M/C N/B] [M/C N/C]}
N<=[M/N N/N] })

(3cond2(M/{C B A N} N/{C B A N} => O/{C B A N})
O/{A<={[M/A N/A] [M/B N/A] [M/C N/A]}
B<={[M/A N/B] [M/B N/B] [M/C N/B]}
C<={[M/A N/C] [M/B N/C] [M/C N/C]}
N<=[M/N N/N] })

(3condexamp1(S/{C B A N} T/{C B A N}=> out/[Z Y]/{C B A N})
3cond1(S T => out/Y)
3cond2(S T => out/Z))

Expressions **3cond1** and **3cond2** express the mappings of the interaction behaviors that realize the interaction mapping. The expression **3condexamp** referencing **3cond1** and **3cond2** expresses how the mappings dependently associate to realize the interaction of **S/** and **T/** to **out/[Y Z]/**.

Again, all possible interaction mappings can be expressed by arranging the output differentnesses in the mapping tables of the association interaction behaviors to represent any desired mapping to the associated output conditions.

There is less condition differentiation and more association differentiation in that now the association differentness of **S** and **T** are necessary to differentiate the presented inputs and **out/X** and **out/Y** must be associationaly ordered to differentiate **out/Y/A out/Z/B** from **out/Y/B outZ/A**.

5.6.2. Second: only five interaction mappings available

Condition differentiation is further constrained by limiting the available association behavior interaction mappings to only five given mappings that cannot be rearranged. The example interaction mapping remains identical to Figure 5.8 but realizing the mapping behavior changes dramatically.

To this point in the walk along the spectrum the ability to map any presented input to any desired output has been supported by the availability of black boxes of all possible association behavior interaction mappings of the available differentness conditions. How the recognition of the presented input and the mapping to assertion of the corresponding output happens within the black box of the association interaction behavior has not been expressed. With the limitation of interaction mappings the internal mechanics of this black box mapping are no longer universally available for all possible mappings but only for the limited mappings. Now the generality of mapping behavior must be explicitly expressed in terms of association relations among the limited available mappings.

5.6.2.1. Recognition of presented input.

The recognition of presented input differentness is accomplished with two association behavior mappings, Equality and Rotate shown in Figure 5.10.

The **Equality** behavior expresses the recognition of a specific presentation of interaction differentness conditions, **CC**. If the input to **Equality** is **CC** the behavior transitions its output to condition **C** indicating equality success. For any other combination of presented interaction differentness conditions it transitions its output to condition **A** indicating equality failure.

The **Rotate** behavior rotates a presented input interaction differentness condition to a next interaction differentness condition in order. A sequence of rotates will rotate a presented input interaction differentness condition through all of the possible interaction differentness conditions. One of these rotations will output the specific equality interaction differentness condition **C**.

Condition **N** plays directly through the **Rotate** behavior and condition **NN** plays directly through the **Equality** behavior.

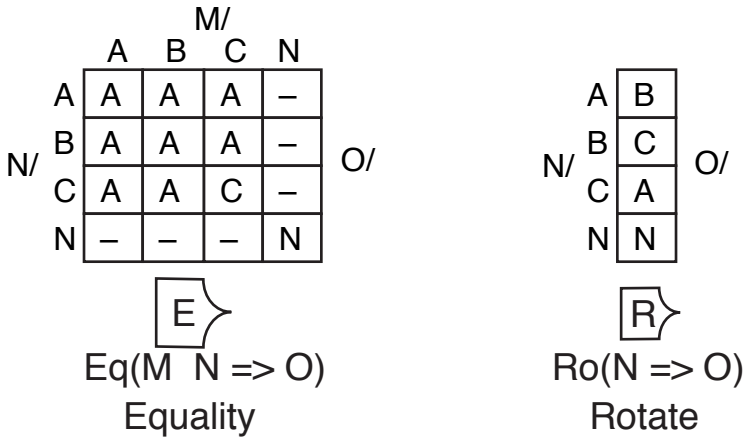


Figure 5.10. The interaction mappings for equality and rotate behaviors.

The interaction mapping expressions.

(Eq(M/{C B A N} N/{C B A N} => O/{C B A N})
O/{A<={[M/A N/A] [M/A N/B] [M/A N/C]
[M/B N/A] [M/B N/B] [M/B N/C]
[M/C N/A] [M/C N/B] [M/C N/C]}
C<= [M/C N/C]
N<=[M/N N/N] })

(Ro(N/{C B A N} => O/{C B A N})
O/{A<= N/C
B<=N/A
C<=N/B
N<=N/N })

5.6.2.2. The cross association recognition network

The presented input condition can be recognized by how many rotations it takes to rotate the input condition to the specific equality condition **C**. Condition **C** after two rotations means that the input condition is **A**. Condition **C** after one rotation means that the input condition is **B**. Condition **C** after no rotation means that the input condition is **C**. For each input each rotated condition will be asserted at a different place of association. One of the places will assert **C**

indicating the presented input condition. The rotation places of association are cross associated. In the cross association of the rotation places there will be one occurrence of **CC** identifying the presented input conditions. One **Equality** behavior in a rank of **Equality** behaviors will recognize the one occurrence of **CC** and transition its output to **C**. The remaining eight **Equality** behaviors will transition their output to A indicating recognition failure. Presentation **S/N** and **T/N** is recognized with all nine **Equality** behaviors asserting **N**.

The cross association recognition portion of the interaction network is illustrated in Figure 5.11. An interaction begins with an empty network with **S/N** and **T/N** presented as input and all of the association interaction behaviors asserting **N**. S/**A** and T/**B** are presented to the network input. The three conditions are represented in color showing the flow of presented conditions through the rotations, into the rank of equality behaviors and to the assertion of recognition condition **C**. The specific presentation of conditions is recognized at a particular place of association in the network. After the interaction the input presentation transitions to **S/N** and **T/N** and the network empties of interaction conditions.

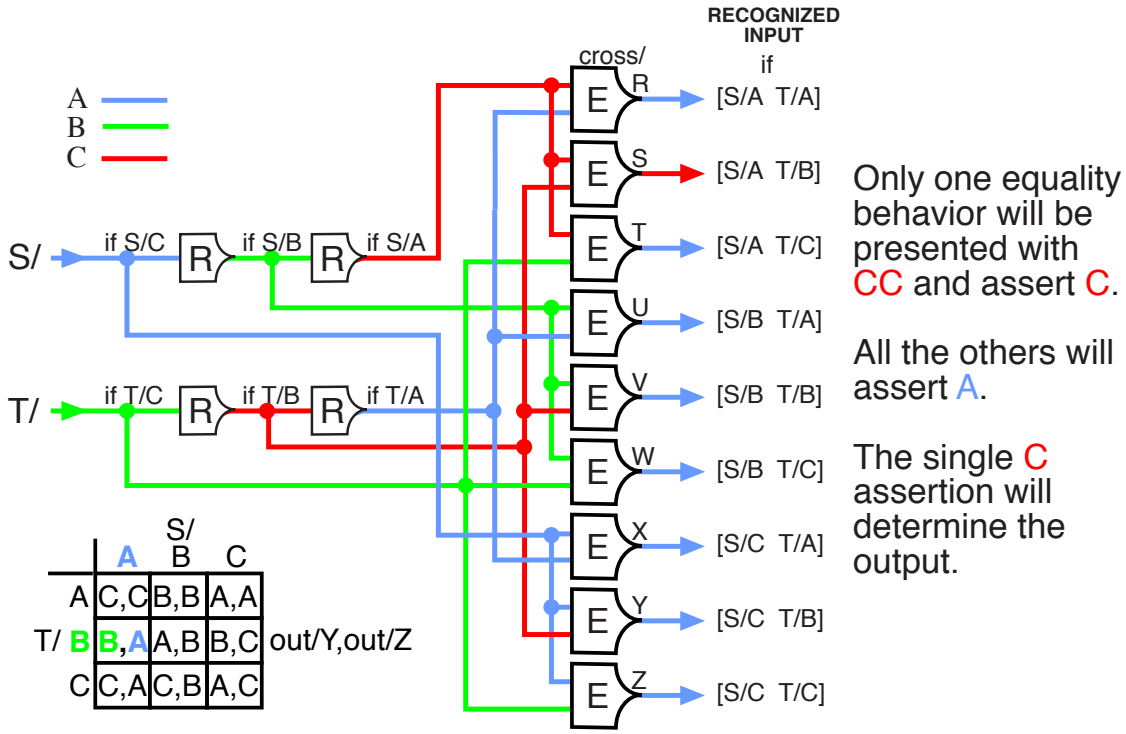


Figure 5.11. Cross association recognition network for example interaction with three differentness conditions and limited behavior mappings.

The cross association recognition portion of the interaction dependency expression.


```
(3condexamp2( S/{C B A N}  T/{C B A N}  => OUT/[Y Z]/{C B A N} )
        ( cross/[R S T U V W X Y Z]/{C B A N} )
/* cross association recognition. Only one transitions to C the other 8 transition to A.*/
/* When both inputs are N all of the equality behaviors will transition to N. */
                cross/[R<=Eq( Ro( Ro( S ))  Ro( Ro( T ))) /* recognize S/A  T/A  */
                       S<=Eq( Ro( Ro( S ))  Ro( T ))      /* recognize S/A  T/B  */
                       T<=Eq( Ro( Ro( S ))  T)            /* recognize S/A  T/C  */
                       U<=Eq( Ro( S )  Ro( Ro( T )))      /* recognize S/B  T/A  */
                       V<=Eq( Ro( S )  Ro( T ))           /* recognize S/B  T/B  */
                       W<=Eq( Ro( S )  T)                 /* recognize S/B  T/C  */
                       X<=Eq( S  Ro( Ro( T )))            /* recognize S/C  T/A  */
                       Y<=Eq( S  Ro ( T ))                /* recognize S/C  T/B  */
                       Z<=Eq( S  T )  ]...                /* recognize S/C  T/C  */.....)
```

5.6.2.3. Assertion of the mapped output

The rank of **equality** behaviors outputs one **C** and eight **A**s. The task is to have the one **C** condition assert the appropriate output condition by prioritizing the flow of **B**s and **C**s over the flow of **A**s to the output assertion. This is accomplished with two **Assert** behaviors and one **Priority** behavior, shown in Figure 5.12.

If the **AssertB** behavior is presented with condition **C** it will output condition **B** but if presented with condition **A** will output condition **A**.

If the **AssertA** behavior is presented with condition **C** it will output condition **A** and if presented with condition **A** will output condition **A**. If the one **C** condition is converted to **A** there are no **B**s or **C**s converging to the output. There are only **A**s converging and the output will be **A**.

Condition **C** does not need to be transitioned into condition **C** so there is no **AssertC** behavior.

The **priority** behavior will pass the highest priority condition with **A** the lowest priority **B** the middle and **C** the highest priority. A network of priority behaviors will pass the highest priority condition presented to the network.

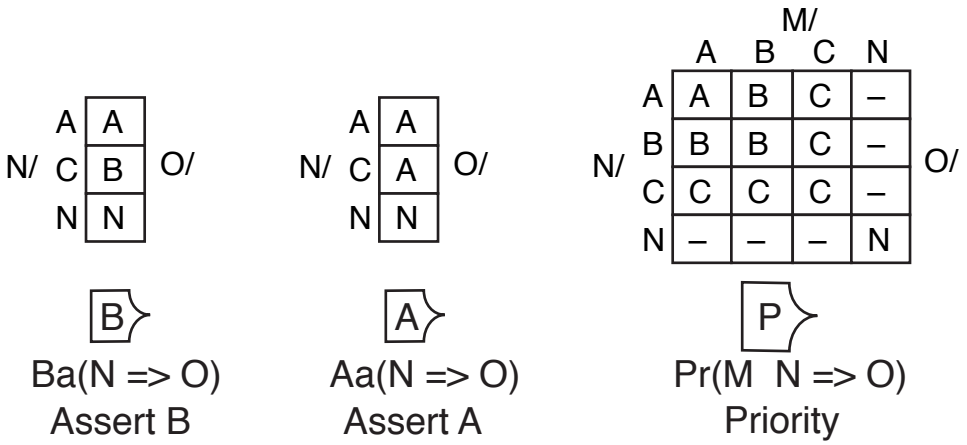


Figure 5.12. Output convergence behaviors.

The interaction mapping expressions:

```
(Pr( M/{C B A N}  N/{C B A N}/ => O/{C B A N} )
            O/{A<=[M/A  N/A]
               B<={[M/A  N/B]  [M/B  N/A]  [M/B  N/B]}
               C<={[M/C  N/A]  [M/C  N/B]  [M/C  N/C]  [M/A  N/C]  [M/B N/C]}
               N<=[M/N  N/N]  } )

(Aa( N/{C B A N}  => O/{C B A N} )
            O/{A<= {N/A  N/C}
               N<=N/N } )

(Ba( N/{C B AN}  => O/{C B A N} )
            O/{A<= N/A
               B<=N/C
               N<=N/N }  )
```

5.6.2.4. The convergence assertion network

The one condition C of the cross association recognition determines the output conditions. For each recognized input that specifies output condition B the recognizing C condition flows through an **AssertB** behavior. For each recognized input that specifies output condition A the recognizing C condition flows through an **AssertA** behavior. Each recognized input that specifies output condition C is not transitioned because the recognition success condition is already C. The result of the assert behaviors is eight A conditions and one asserted output condition which may also be condition A. If all nine recognition conditions are condition **N** then **out/Y** will be condition **N**.

In Figure 5.13 all nine input recognitions flowing through locality **cross/** are presented to a priority network that converges the flows to the assertion of **out/Y**. The three condition are represented in color showing the flow of presented conditions through the assertion and priority behaviors to **out/Y**.

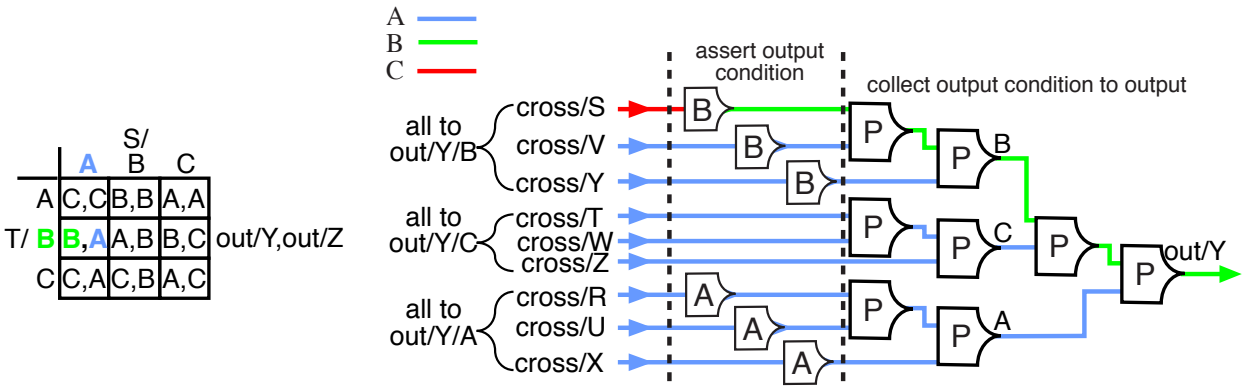


Figure 5.13. Three condition output convergence network to assert out/Y.

Assertion convergence dependency expression:

out/Y<= Pr(Pr(Pr(Pr(Ba(cross/S) Ba(cross/V)) Ba(cross/Y))
Pr(Pr(cross/T cross/W) cross/Z))
Pr(Pr(Aa(cross/R) Aa(cross/U)) Aa(cross/X)))

5.6.2.5. The complete network realizing the example interaction.

Figure 5.14 shows the association network expressing the example interaction with three differentness conditions **A**, **B** and **C** and the set of five interaction behaviors **Ro(…)**, **Eq(…)**, **Ba(…)**, **Aa(…)** and **Pr(…)** .

An interaction begins with the network empty of interaction condition with **S/N T/N,** all the internal behaviors asserting **N** and **out/Y/N** and **out/Z/N.** The inputs transition to **S/A** and **T/B**. The three condition are represented in color showing the flow of presented conditions through the rotations, into the rank of equality behaviors and through the convergence network to **out/Y** and **out/Z**. When **S/** and **T/** transition to **N**, all internal behaviors transition to **N** and **out/Y** and **out/Z** transitions to **N** leaving the interaction network empty with completely **N** bounding one temporal instance of one specific interaction of differentness.

The flow through the network is a collaboration between differentness of condition, differentness of interaction behavior and differentness of place of association in the network.

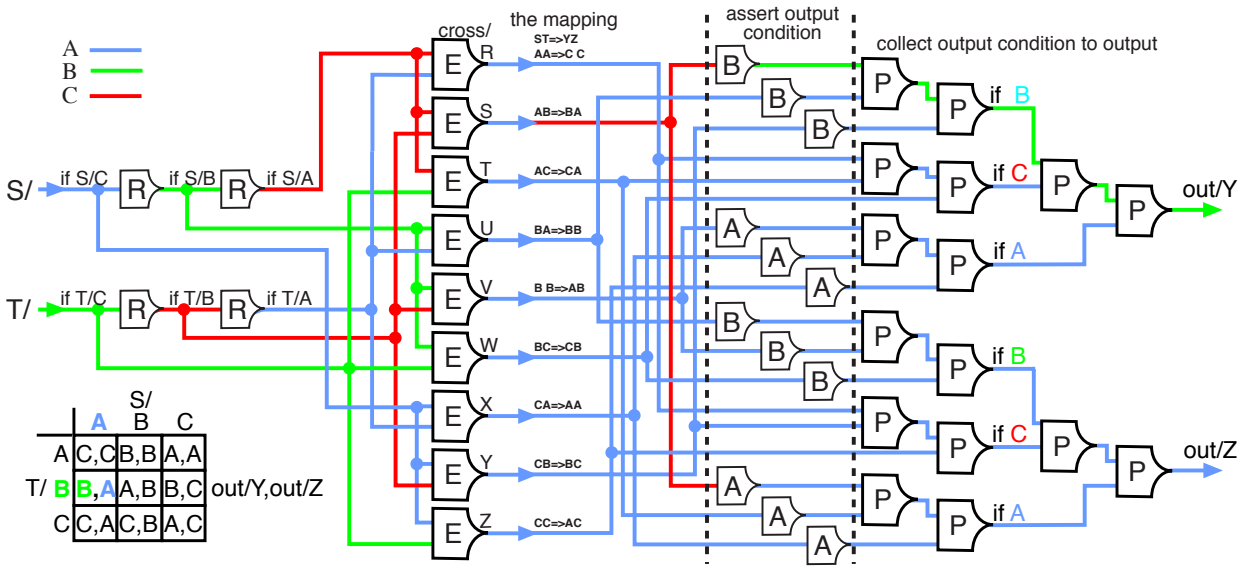


Figure 5.14. Three differentness conditions with five interaction mappings realizing the example interaction with 41 behaviors.

The complete interaction network expression.

```
(3condexamp2( S/{C B A N}  T/{C B A N}  => OUT/[Y Z]/{C B A N} )
        ( cross/{R S T U V W X Y Z}/{C B A N} )
/* cross association recognition. Only one transitions to C the other 8 transition to A.*/
/* When both inputs are N all of the equality behaviors will transition to N. */
                cross/[R<=Eq( Ro( Ro( S ))  Ro( Ro( T ))) /* recognize S/A  T/A  */
                        S<=Eq( Ro( Ro( S ))  Ro( T ))      /* recognize S/A  T/B  */
                        T<=Eq( Ro( Ro( S ))  T)            /* recognize S/A  T/C  */
                        U<=Eq( Ro( S )  Ro( Ro( T )))      /* recognize S/B  T/A  */
                        V<=Eq( Ro( S )  Ro( T ))           /* recognize S/B  T/B  */
                        W<=Eq( Ro( S )  T)                 /* recognize S/B  T/C  */
                        X<=Eq( S  Ro( Ro( T )))            /* recognize S/C  T/A  */
                        Y<=Eq( S  Ro ( T ))                /* recognize S/C  T/B  */
                        Z<=Eq( S  T )  ]…                  /* recognize S/C  T/C  */…
/* The convergence to out/Y */
        out/Y<= Pr( Pr( Pr( Pr( Ba( cross/S) Ba( cross/V)) Ba( cross/Y))  /* assert Y/B */
                        Pr( Pr( cross/T  cross/W) cross/ZZ))             /* assert Y/C */
                        Pr( Pr( Aa( cross/R) Aa( cross/U)) Aa( cross/X))) /* assert Y/A */
/* The convergence to out/Z  */
        out/Z<= Pr( Pr( Pr( Pr( Ba( cross/U) Ba( cross/V)) Ba( cross/W))   /* assert Z/B */
                        Pr( Pr( cross/R  cross/Y) cross/Z))                /* assert Z/C */
                        Pr( Pr( Aa( cross/S) Aa( cross/T)) Aa( cross/X)))  /* assert Z/A */ )
```

All possible interaction mappings are now expressed by restructuring the network in terms of the limited mappings. There is less condition differentiation and considerably more association differentiation.

5.7. Constrained to two available differentness conditions

Constrained to two available interaction differentness conditions named **A** and **B** the inputs with three differentnesses and the output with nine differentnesses have to be represented with two interaction differentness conditions and four interaction differentness conditions respectively as shown in Figure 5.15.

The interaction mapping.

mapping table

T/Y, T/Z \ S/Y, S/Z	A,A	A,B	B,A
A,A	A,A,A,A	A,A,A,B	A,A,B,A
A,B	A,A,B,B	A,B,A,A	A,B,A,B
B,A	A,B,B,A	A,B,B,B	B,A,A,A

out/W, out/X, out/Y/, out/Z/

specified interaction relations

```
[S/Y/A  S/Z/A  T/Y/A  T/Z/A]  => [out/Y/A  out/Z/A  out/Y/A  out/Z/A]
[S/Y/A  S/Z/B  T/Y/A  T/Z/A]  => [out/Y/A  out/Z/A  out/Y/A  out/Z/B]
[S/Y/B  S/Z/A  T/Y/A  T/Z/A]  => [out/Y/A  out/Z/A  out/Y/B  out/Z/A]
[S/Y/A  S/Z/A  T/Y/A  T/Z/B]  => [out/Y/A  out/Z/A  out/Y/B  out/Z/B]
[S/Y/A  S/Z/B  T/Y/A  T/Z/B]  => [out/Y/A  out/Z/B  out/Y/A  out/Z/A]
[S/Y/B  S/Z/A  T/Y/A  T/Z/B]  => [out/Y/A  out/Z/B  out/Y/A  out/Z/B]
[S/Y/A  S/Z/A  T/Y/B  T/Z/A]  => [out/Y/A  out/Z/B  out/Y/B  out/Z/A]
[S/Y/A  S/Z/B  T/Y/B  T/Z/A]  => [out/Y/A  out/Z/B  out/Y/B  out/Z/B]
[S/Y/B  S/Z/A  T/Y/B  T/Z/A]  => [out/Y/B  out/Z/A  out/Y/A  out/Z/A]
```

Figure 5.15. Interaction mapping with two available differentness conditions.

5.7.1. Two differentness condition interaction behavior mappings

With only two interaction differentness conditions **B** becomes the Equality condition and **AssertB** is no longer needed. The limited interaction mappings become the four association interaction behaviors in Figure 5.16.

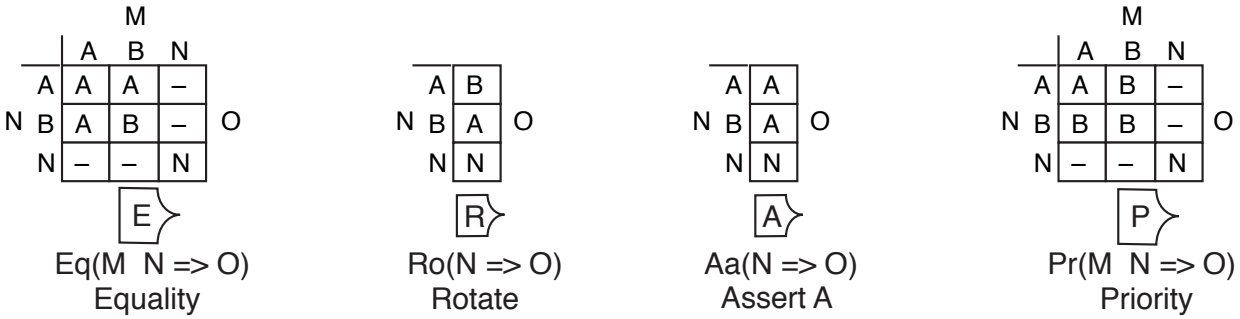


Figure 5.16. Two differentness condition association interaction behaviors.

The component behavior expressions:

```
(Eq( M/{B A N}  N/{B A N}  => O/{B A N} )
            O/{A<={[M/A  N/A]  [M/A  N/B]  [M/B  N/A]}
               B<=  [M/B  N/B]
               N<=[M/N  N/N]  } )
```


(Ro(N/{B A N} => O/{B A N})
O/{A<= N/B
B<=N/A
N<=N/N })

(Aa(N/{B A N} => O/{B A N})
O/{A<= {N/A N/B}
N<=N/N })

(Pr(M/{B A N} N/{B A N}/ => O/{B A N})
O/{A<=[M/A N/A]
B<= {[M/A N/B] [M/B N/A] [M/B N/B]}
N<=[M/N N/N] })

5.7.2. Recognition of presented input

The differentness localities **S** and **T** are each represented as two places of association **S/Y/**, **S/Z/** and **T/Y/**, **T/Z/** extending the two condition differentnesses to cover the three input interaction differentnesses. With the input differentnesses represented with two localities two stages of cross association recognition are required. In the left half of Figure 5.17 each input locality is expanded with **rotate** behavior presented to a rank of three **equality** behaviors only one of which will assert **B** and the rest will assert **A**. The two resulting localities of asserted conditions are then cross associated and only one cross association will be the specific recognition presentation of **BB**. One of the rank of **equality** behaviors will recognize the one cross association presenting **BB** asserting its output condition **B** indicating recognition success with the other eight **equality** behaviors assert output condition **A** indicating recognition failure.

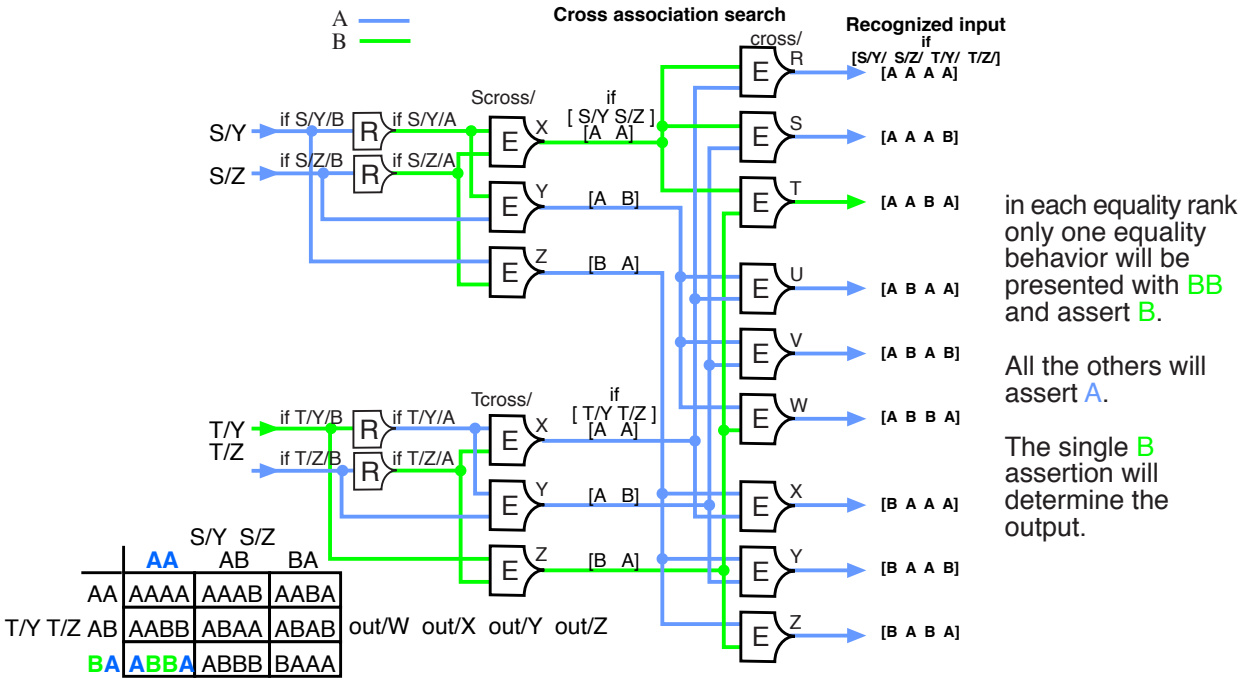


Figure 5.17. Cross association recognition network for example interaction with two differentness conditions.

The recognition portion of the interaction expression:

```
(2condexamp(S/[Y Z]/{B A N}  T/[Y Z]/{B A N}  => OUT/[W X Y Z]/{B A N})
      (cross/[R S T U V W X Y Z]/{B A N}   Scross/[X Y Z]/{B A N}
        Tcross/[X Y Z]/{B A N} )
          Scross/[X<=Eq( Ro(S/Y)  Ro(S/Z) )          /* recognize S/Y/A  S/Z/A  */
                  Y<=Eq( Ro(S/Y)  S/Z )              /* recognize S/Y/A  S/Z/B  */
                  Z<=Eq( S/Y  Ro(S/Z) )  ]           /* recognize S/Y/B  S/Z/A  */
          Tcross/[X<=Eq( Ro(T/Y)  Ro(T/Z) )          /* recognize T/Y/A  T/Z/A  */
                  Y<=Eq( Ro(T/Y)  T/Z )              /* recognize T/Y/A  T/Z/B  */
                  Z<=Eq( T/Y  Ro(T/Z) )  ]           /* recognize T/Y/B  T/Z/A  */
/* cross association recognition. Only one transitions to B the other 8 transition to A.*/
/* When the input is all N all of Scross, all of Tcross and all of cross transition to N  */
     cross/[R<=E( Scross/X  Tcross/X)     /* recognize S/Y/A  S/Z/A  T/Y/A  T/Z/A */
             S<=Eq( Scross/X  Tcross/Y)   /* recognize S/Y/A  S/Z/A  T/Y/A  T/Z/B */
             T<=Eq( Scross/X  Tcross/Z)   /* recognize S/Y/A  S/Z/A  T/Y/B  T/Z/A */
             U<=Eq( Scross/Y  Tcross/X)   /* recognize S/Y/A  S/Z/B  T/Y/A  T/Z/A */
             V<=Eq( Scross/Y  Tcross/Y)   /* recognize S/Y/A  S/Z/B  T/Y/A  T/Z/B */
             W<=Eq( Scross/X  Tcross/Z)   /* recognize S/Y/A  S/Z/B  T/Y/B  T/Z/A */
             X<=Eq( Scross/Z  Tcross/X)   /* recognize S/Y/B  S/Z/A  T/Y/A  T/Z/A */
             Y<=Eq( Scross/Z  Tcross/X)   /* recognize S/Y/B  S/Z/A  T/Y/A  T/Z/B */
             Z<=Eq( Scross/Z  Tcross/Z) ] /* recognize S/Y/B  S/Z/A  T/Y/B  T/Z/A */
 ……)
```


5.7.3. Assertion of the mapped output

The one condition **B** of the recognition determines each output through a **priority** convergence network in which the recognition results are first **priority** collected and then presented to the **AssertA** behavior just before the final **priority** interaction behavior that asserts the output condition.

The one **B** condition and the eight **A** conditions from the cross association recognition will pass through the convergence network and set **out/Z/A** or **out/Z/B**. When all nine inputs are **N** the output will be **out/Z/N**.

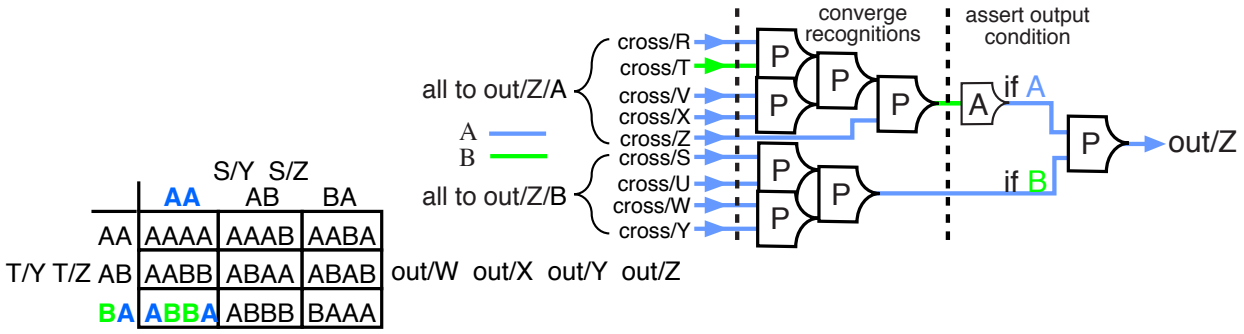


Figure 5.18. Two condition output convergence network to assert out/Z.

Output convergence portion of interaction dependency expression:

out/Z<= Pr(Aa(Pr(Pr(Pr(cross/A cross/C) Pr(cross/E cross/G)) cross/I))
Pr(Pr(cross/B cross/D) Pr(cross/F cross/H)))

5.7.4. The complete network realizing the example interaction.

An interaction begins as an empty network with **S/Y/N**, **S/Z/N**, **T/Y/N**, **T/Z/N** all internal behaviors asserting **N** and **out/W/N**, **out/X/N**, **out/Y/N**, **out/Z/N.** Inputs **S/Y/A**, **S/Z/A**, **T/Y/B**, **Y/Z/A** transition to interaction differentness conditions and the output localities **out/W/A**, **out/X/B**, **out/Y/B**, **out/Z/A** transitions to the determined interaction differentness conditions. **S/Y**, **S/Z**, **T/Y**, **Y/Z** transition back to **N**, and **out/W**, **out/X**, **out/Y**, **out/Z** transitions to **N** leaving the interaction network empty of interaction differentness conditions bounding one temporal instance of one specific interaction of differentness.

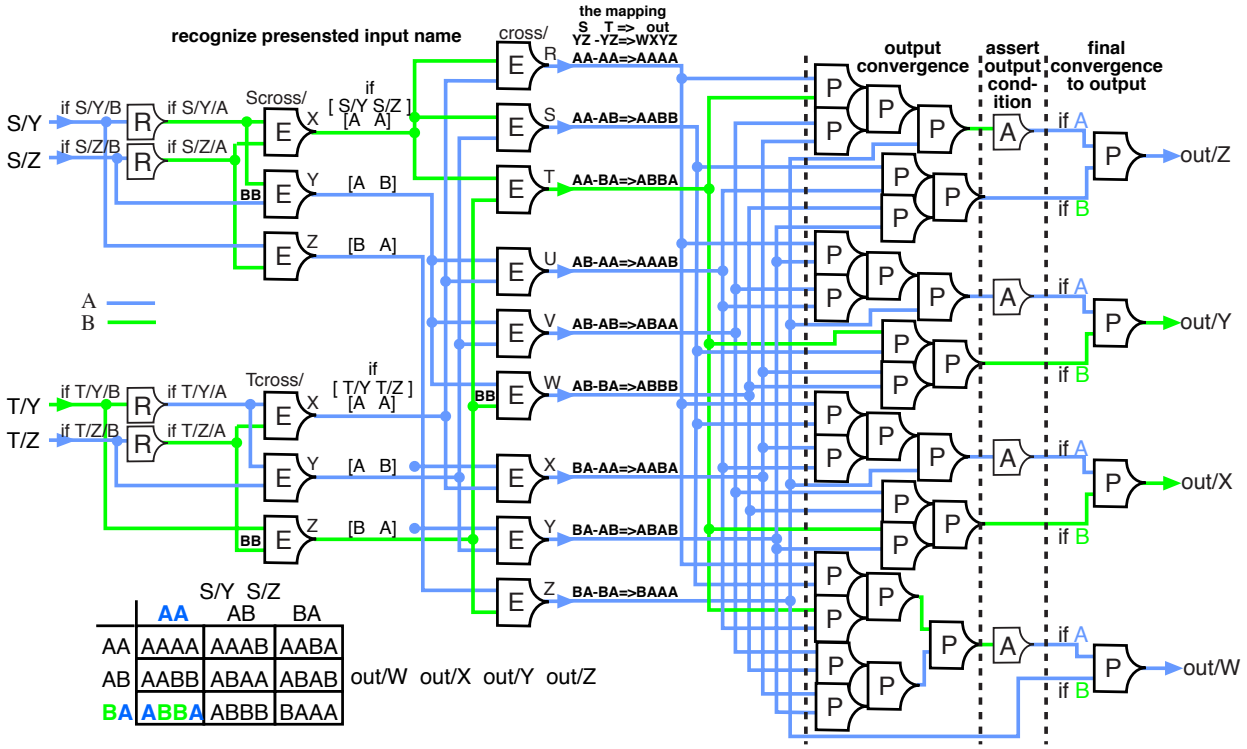


Figure 5.19. Two interaction differentness conditions with four interaction mappings realizing the example interaction with 55 behaviors.

The complete interaction dependency expression:

```
(2condexamp( S/[Y Z]/{B A N}  T/[Y Z]/{B A N}  => OUT/[W X Y Z]/{B A N} )
        ( cross/[R S T U V W X Y Z]/{B A N}
                Scross/[X Y Z]/{B A N} Tcross/[X Y Z]/{B A N} )
              Scross/[X<=Eq( Ro(S/Y)  Ro(S/Z) )          /* recognize S/Y/A  S/Z/A  */
                      Y<=Eq( Ro(S/Y)  S/Z )              /* recognize S/Y/A  S/Z/B  */
                      Z<=Eq( S/Y  Ro(S/Z) )  ]           /* recognize S/Y/B  S/Z/A  */
             Tcross/[X<=Eq( Ro(T/Y)  Ro(T/Z) )           /* recognize T/Y/A  T/Z/A  */
                      Y<=Eq( Ro(T/Y)  T/Z )              /* recognize T/Y/A  T/Z/B  */
                      Z<=Eq( T/Y  Ro(T/Z) )  ]           /* recognize T/Y/B  T/Z/A  */
/* cross association recognition. Only one transitions to B the other 8 transition to A.*/
/* When the input is all N all of Scross, all of Tcross and all of cross transition to N  */
      cross/[R<=Eq( Scross/X  Tcross/X)     /* recognize S/Y/A  S/Z/A  T/Y/A  T/Z/A */
             S<=Eq( Scross/X  Tcross/Y)     /* recognize S/Y/A  S/Z/A  T/Y/A  T/Z/B */
             T<=Eq( Scross/X  Tcross/Z)     /* recognize S/Y/A  S/Z/A  T/Y/B  T/Z/A */
             U<=Eq( Scross/Y  Tcross/X)     /* recognize S/Y/A  S/Z/B  T/Y/A  T/Z/A */
             V<=Eq( Scross/Y  Tcross/Y)     /* recognize S/Y/A  S/Z/B  T/Y/A  T/Z/B */
             W<=Eq( Scross/Y  Tcross/Z)     /* recognize S/Y/A  S/Z/B  T/Y/B  T/Z/A */
             X<=Eq( Scross/Z  Tcross/X)     /* recognize S/Y/B  S/Z/A  T/Y/A  T/Z/A */
             Y<=Eq( Scross/Z  Tcross/Y)     /* recognize S/Y/B  S/Z/A  T/Y/A  T/Z/B */
             Z<=Eq( Scross/Z  Tcross/Z) ]   /* recognize  S/Y/B  S/Z/A  T/Y/B  T/Z/A */
/* map the recognition to the output */
        out/[Z<= Pr( Aa( Pr( Pr( Pr( cross/R  cross/T) Pr( cross/V  cross/X)  ) cross/Z) )
                         Pr( Pr( cross/S  cross/U) Pr( cross/W cross/Y))   )
             Y<= Pr( Aa( Pr( Pr( Pr( cross/R  cross/U) Pr( cross/V  cross/Y)  ) cross/Z) )
                         Pr( Pr( cross/S  cross/T) Pr( cross/W cross/X))   )
             X<= Pr( Aa( Pr( Pr( Pr( cross/R  cross/S) Pr( cross/U  cross/X)  ) cross/Z) )
                         Pr( Pr( cross/T  cross/V) Pr( cross/W cross/Y))   )
            W<= Pr( Aa( Pr( Pr( Pr( cross/R  cross/S) Pr( cross/T  cross/U)  )
                            Pr( Pr( cross/V  cross/W) Pr( cross/X  cross/Y)  )) )
                   cross/Z )  ]  )
```

There is less condition differentiation and more association differentiation.

5.7.5. Two differentness condition interaction with extrinsic coordination

With an extrinsic coordinating behavior such as a mathematician or timing analysis and a clock the **N** condition is no longer needed and the **AssertA** behavior, which becomes a default, is no longer needed. Two differentness condition extrinsically coordinated interaction behaviors reduce to Boolean logic as shown in Figure 5.20.

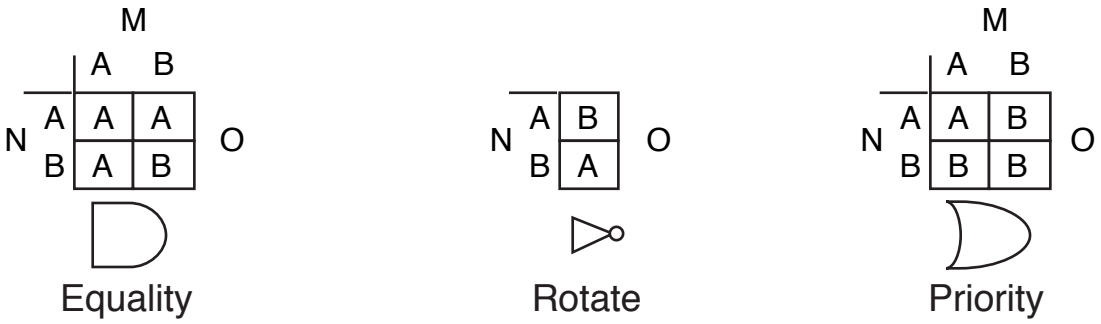


Figure 5.20. Two condition interaction behaviors assuming extrinsic coordination.

An extrinsically coordinated network can make assumptions about behavior such as that after a specific time interval all interactions within the association network have occurred. With only two possibilities and after the interval one possibility was not successful it can be assumed by default that the other possibility was successful. At a specified time or at the discretion of the mathematician a so called truth function will be asserting T indicating its own truth or it will be asserting F indicating by default the alternate truth. The converge network for each output with default behavior reduces to Figure 5.21.

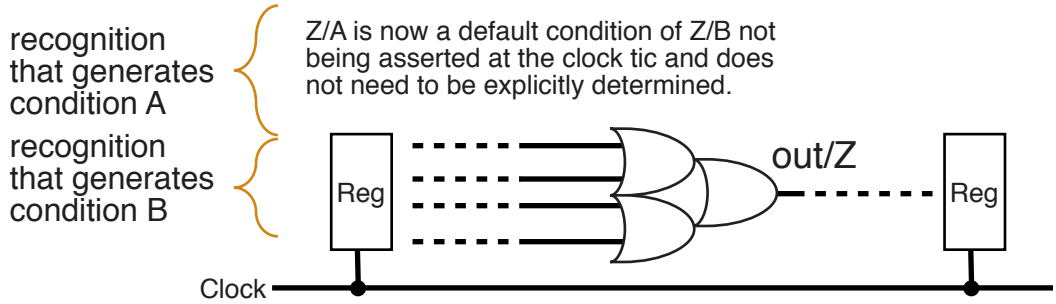


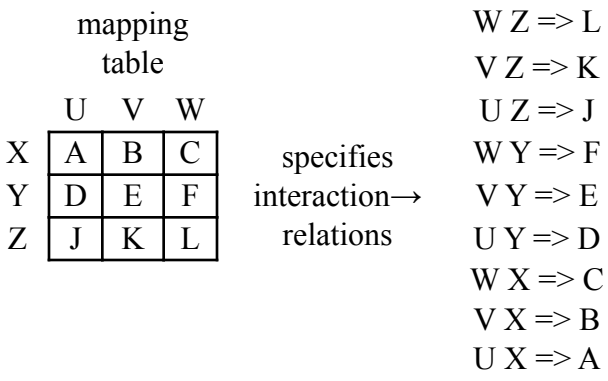
Figure 5.21. Extrinsically coordinated output convergence.

Also, with the truth function comes the "don't care condition. If a truth function can perform its duty by ignoring a related variable then it does not care what the condition of the variable might be. With extrinsic control logical completeness is no longer necessary.

Figure 5.21 seems to be much more efficient than Figure 5.18 but it comes with the conceptual cost of an incomplete logic and a logically extrinsic critical timing requirement and timing referent. Extrinsic coordination is a crutch propping up an insufficient logic.

5.8. Constrained to one differentness condition: Pure association differentiation

Constrained to one interaction differentness condition named **A** condition interaction differentiation ceases to exist and all interaction differentiation is in terms of differentness of place of association. The mapping is now entirely in terms of names of places of association rather than in terms of names of conditions. The collaboration of condition differentnesses **A** and **N** reduce to serving only to assert or not assert the differentness of a place of association.



There is no longer any need for the **Rotate** behavior or the **Assert** behavior. There are no interactions of different conditions. There are only interactions of two or more **A**s which can be determined with a threshold. The Equality behavior reduces to an "all of" behavior. The priority behavior reduces to a "one of" behavior. Replacing **A** with **D** the behaviors of Figure 5.22 correspond to the "all of" and "one of" behaviors of Section 3.2. For continuity this chapter will continue using **A** as the only differentness condition. These primitive behaviors do not have reference names but are referenced syntactically with [ ] and { }.

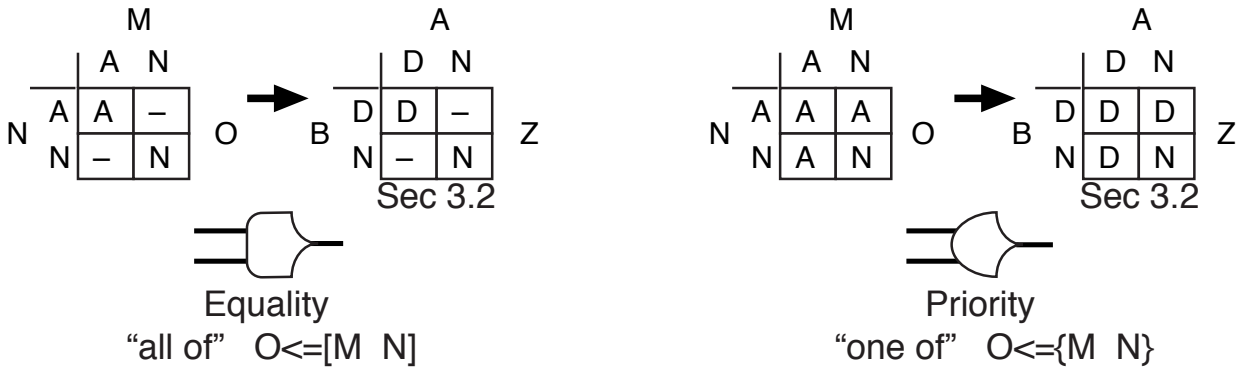


Figure 5.22. One differentness condition association interaction behaviors.

Mutual exclusivity, previously represented by asserting one at a time of two or more differentness conditions at a single place of association is now represented with two or more places of association only one of which, at a time, will transition to **A** with the rest remaining at **N**. In Figure 5.23 the locality named **S** consists of three places of association named **U**, **V** and **W** only one of which, at a time, will transition to **A**. The locality named **T** consists of three places of association named **X**, **Y** and **Z** only one of which, at a time, will transition to **A**. The locality named **out** consists of nine places of association named **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H** and **I** only one of which, at a time, will transition to **A**.

An interaction will begin with the presented input at **N** and with all behaviors asserting **N**. Only one of **U**, **V** or **W** and one of **X**, **Y** or **Z** will transition to **A.** The other inputs will remain at **N**. Only one of the **equality** behaviors will be presented with input completeness **AA** and transition its output to **A**. The other eight **equality** behaviors will be presented with incomplete input and will not transition their outputs which will remain asserting **N**. When the presented inputs transition back to all **N** the one **equality** behavior will transition its output to **N** and all of the **equality** behaviors will be asserting **N** leaving the interaction network empty and bounding

one temporal instance of one specific interaction of differentness. Figure 5.23 illustrates the transition of input differentnesses to **S/U/A** and **T/Y/A**.

Two mutually exclusive differentnesses are mutually inclusively cross associated to produce one mutually exclusive differentness.

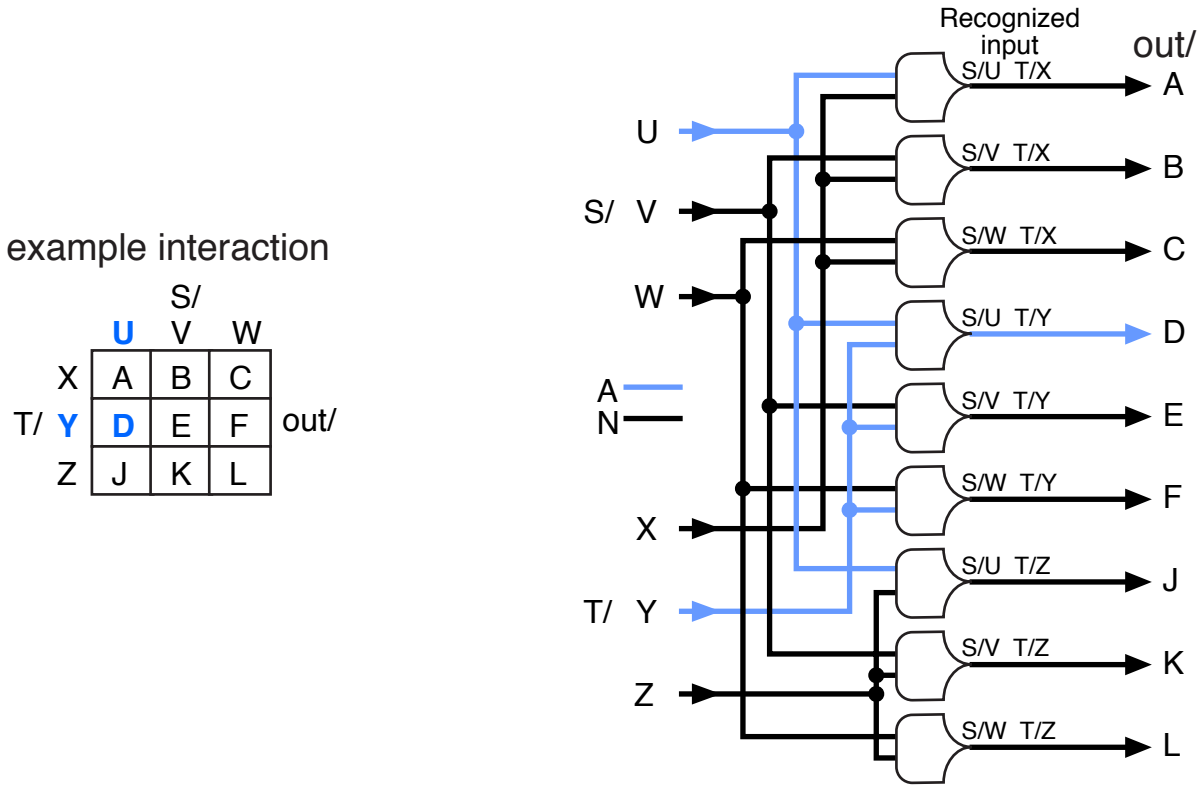


Figure 5.23. One differentness condition with two interaction mappings realizing the example interaction with 9 behaviors.

The complete interaction expression:

(Onecondexamp1 (S/{U,V,W}/{A N}, T/{X,Y,Z}/{A N} -> out/{A,B,C,D,E,FJ,K,L}/{A N});
out/{A<=[S/U T/X]
B<=[S/V T/X]
C<=[S/W T/X]
D<=[S/U T/XY]
E<=[S/V T/Y]
F<=[S/W T/Y]
G<=[S/U T/Z]
H<=[S/V T/Z]
I<=[S/W T/Z] })

There is no condition interaction differentiation and all interaction differentiation is association differentiation.

5.8.1.1. Generality of mapping

The rank of **“all of”** behaviors is the cross association recognition of presented input. The one transition to **A** can realize any mapping to any output locality of fewer than nine differentnesses with convergence through **“one of”** behaviors as illustrated inFigure 5.24. The places of association are renamed with **A**, **B** and **C** to illustrate the differentness of place of association. Differentnesses **S/B/A T/B/A** and **out/B/A** all asserting **/B/A** are all different by virtue of the differentness of the localities of association of **S**, **T** and **out**.

Figure 5.24 illustrates the presentation transition to **S/C/A** and **T/A/A** leading to the assertion of output differentness **out/C/A**.

example interaction

T/ \ S/	A	B	C
A	A	B	C
B	B	C	B
C	A	A	C

out/

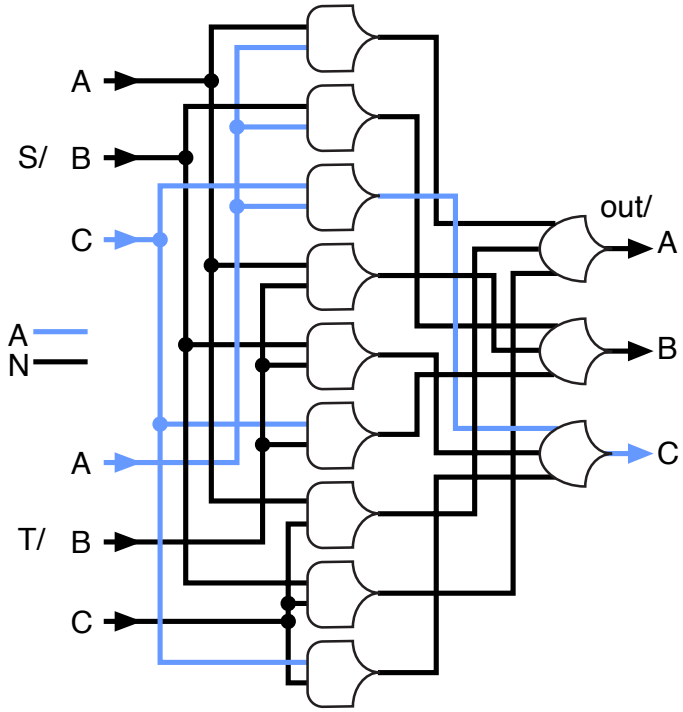


Figure 5.24. General mapping with “all of” and “one of” behaviors.

The complete interaction expression:

(Onecondexamp2 (S/{A B C}/{A N} T/{A B C}/{A N} -> out/{A B C}/{A N})
out/{A<= { [S/A T/A] [S/A T/C] [S/B T/C] }
B<= { [S/B T/A] [S/A T/B] [S/C T/B] }
C<= { [S/C T/A] [S/B T/B] [S/C T/C] } })

Expressional generality, the ability to map any presented input differentness to any asserted output differentness is now realized entirely in terms of association differentiation. There is no differentiation in terms of differentness of condition.

5.9. Spectrum summary: The differentness of differentness

Differentnesses and their interaction can be expressed as differentness of conditions with specific interaction propensities promiscuously associating at a place of common association (pure condition differentiation) or as differentness of places of association with specific association relations each asserting only two different conditions, **A** and **N** with promiscuous interaction propensities (pure association differentiation) or with varying proportions of

condition differentiation and of association differentiation across a spectrum of differentiation. There must always be a little bit of each domain of differentiation in the expression of any interaction, at least one place of association with pure condition differentiation and at least two conditions with pure association differentiation.

As the spectrum was traversed from the pure condition differentiation end of the spectrum association differentiation took on increasing expressional duties until with pure association differentiation there was no longer any condition differentiation.

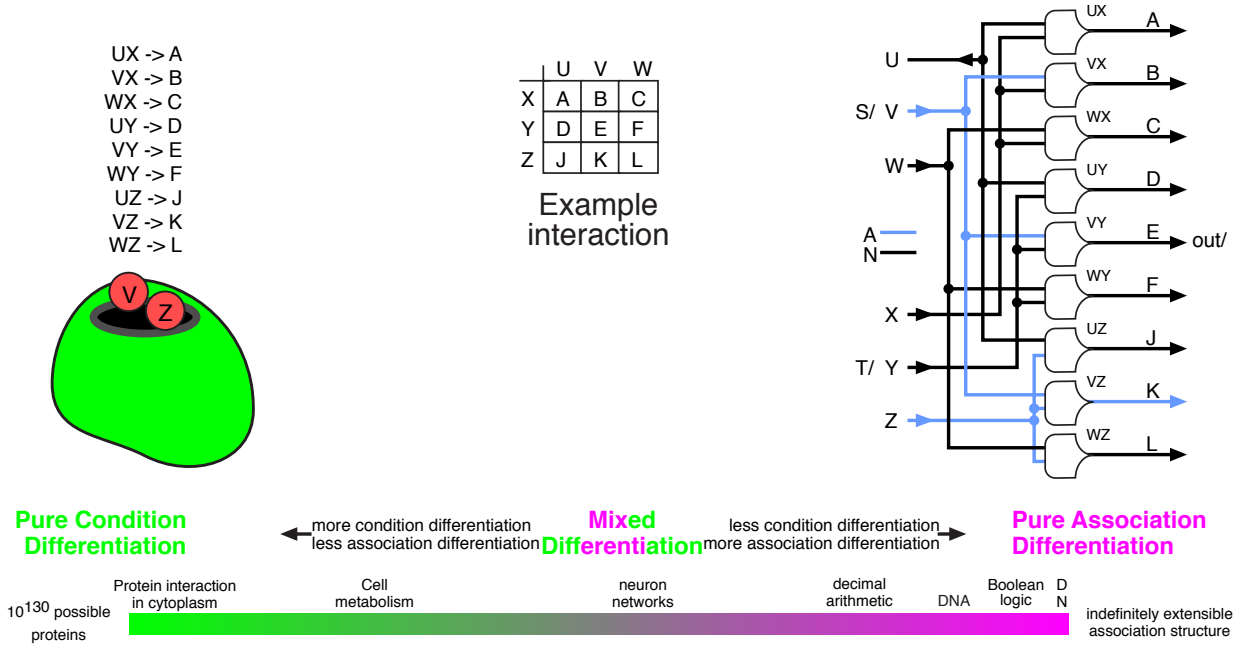


Figure 5.25. The spectrum of differentiation.

In the center of Figure 5.25 is the example interaction. To the right is shown the pure association expression of the interaction. To the left is shown the pure condition expression of the interaction. The pure condition expression and the pure association expression can be viewed as duals in specific respects.

- Each maps directly from the example interaction table.
- There are nine condition interaction propensity relations on the left and nine interaction behaviors on the right.
- There are six input conditions on the left and six input localities on the right.
- There are nine output conditions on the left and nine output localities on the right.
- On the left persistences asserting conditions freely associating in a shaking bag interact according to their asserted condition's interaction propensities.
- On the right a network of statically associated interaction behaviors interact according to their association relations.

The two ends of the spectrum are the expression realms of nature. At the pure condition end is the biological cell with lots of unique conditions, proteins, with specific interaction propensities and relatively little association structure. There are 10^{130} possible protein folding conditions and uncountable possible protein interaction relations. At the pure association end of

the spectrum is the neural network which is the direct association of differentnesses through association interaction behaviors.

The two realms of differentness are coextensive in that a range of differentness of condition reuses and extends the range of differentness expression of a place of association and each place of association reuses and extends the differentness expression of a range of differentness conditions. They collaborate with each extending the expression of differentness of the other to indefinitely extend expression of differentness.

The two realms cooperate in complex ways. A biological cell full of protein conditions contains associating organelles each isolating a pure condition expression. Cells associate to form organs which associate to form an organism in the midst of which is the blood stream, a pure condition expression indiscriminately associating to every cell in the organism.

Humans tend to express inside the spectrum encoding differentness with place-value numbers, place being association differentness and value being condition differentness. There are few conditions, few interaction behaviors and large indefinitely extensible association networks.

One domain of expression can be held constant while the other domain is allowed free rein. Conditions and interaction behaviors can be held constant while supporting arbitrarily complex association networks. Ten different numerals and their arithmetic behaviors support a vast realm of arbitrarily extensible numbers and their interactive association. The association relations can be held constant while condition differentiation is arbitrarily available. All mammals have essentially a constant association structure with the same body design, organ structures and cell structures. The variability among mammals is largely a matter of protein conditions expressed by DNA.

The spectrum of differentiation encompasses and unites disparate expressions of differentness and its interaction that previously appeared to be distinct or only superficially related. Nature and humans interact in fundamentally the same way in terms of differentness and the specific interaction of differentness.

Evolution

At each end of the spectrum the expression of complexity is constrained. The pure condition end of the spectrum is determined by the existing differentness conditions. There is no static structure to capture evolution The pure association end of the spectrum is determined by the association relations of the primitive behaviors there is no dynamism to facilitate change. Only in the middle of the spectrum is there sufficient variety of differentness and range of expressivity forming a sufficient counterpoint of dynamism and accumulation of substantiveness for evolution to occur. Sufficient expressivity of differentness, of dependency relations.

Appendix A: Blinded by Elegance: The Null Convention Logic Library

The reader may have notice that Null Convention Logic (NCL)[1] and its operator library are not mentioned in the text. There are strategic reasons, conceptual reasons, practical reasons and technical reasons.

A.1. Null Convention Logic

Null Convention Logic was conceived as a threshold logic with hysteresis behavior that monotonically transitioned between “diff” completeness and completely “null” or “not diff”. Because of multi-rail differentness representation complementing is a matter of relabeling rails instead of converting signals so there are no signal conversions in the wavefront path. Wavefront path logic is purely positive. The only signal conversion in NCL is the conversion of the closure of an oscillation.

Threshold logic was academically established and fully characterized. After a thorough review of threshold logic by my colleague Dave Duncan the library of NCL operators was conceived in terms of the threshold functions of four or fewer inputs. It turned out that there was an established mapping between positive Boolean functions of four or fewer inputs and threshold functions of four or fewer inputs. It was considered that this correspondence between positive Boolean functions of four or fewer inputs, threshold functions of four or fewer inputs and the NCL library operators as shown in Figure A.1 was a compelling correspondence of primitivities that would be advantageous.

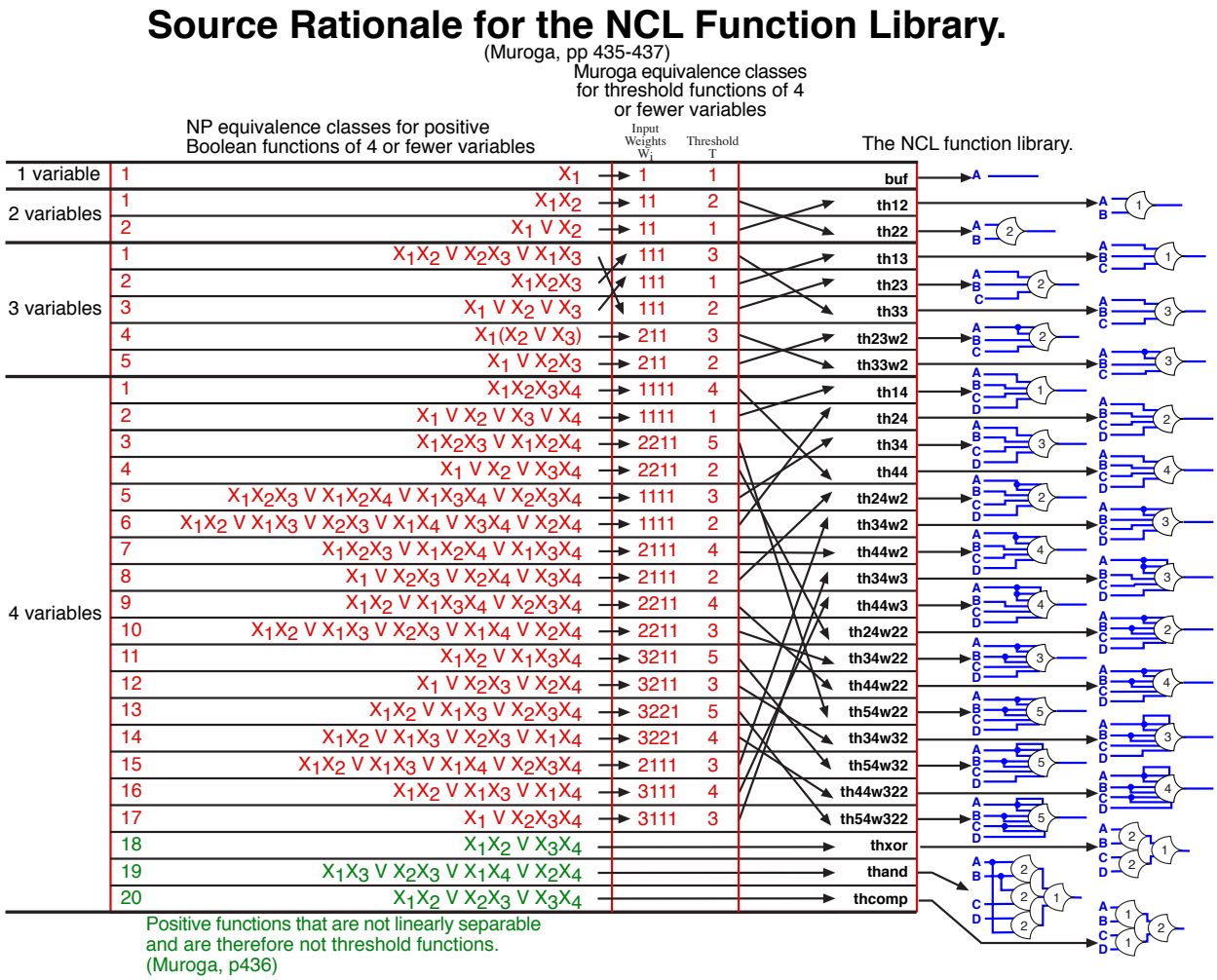


Figure A.1. Function classification rationale for the NCL operator library.

There are 28 PN classes of unate or positive boolean functions of four or fewer variables. There are 25 classes of threshold functions of four or fewer variables. Twenty five of the positive Boolean classes are linearly separable and map directly onto the 25 classes of threshold functions of four or fewer variables. The 3 remaining Boolean functions of four or fewer variables are not linearly separable and do not map onto threshold functions but can map onto threshold circuits. Including these three circuits in the library allowed a complete correspondence between the NCL library and all positive Boolean functions of four or fewer variables as shown in Figure A.2.

We considered this correspondence to be elegant, practical and even necessary to relate to existing practice: a useful strategic positioning.

- We could use Boolean equations to design NCL circuits.
- We could relate to a familiar language.
- We could possibly translate clocked Boolean designs directly to NCL designs.
- We could use standard design tools for NCL.

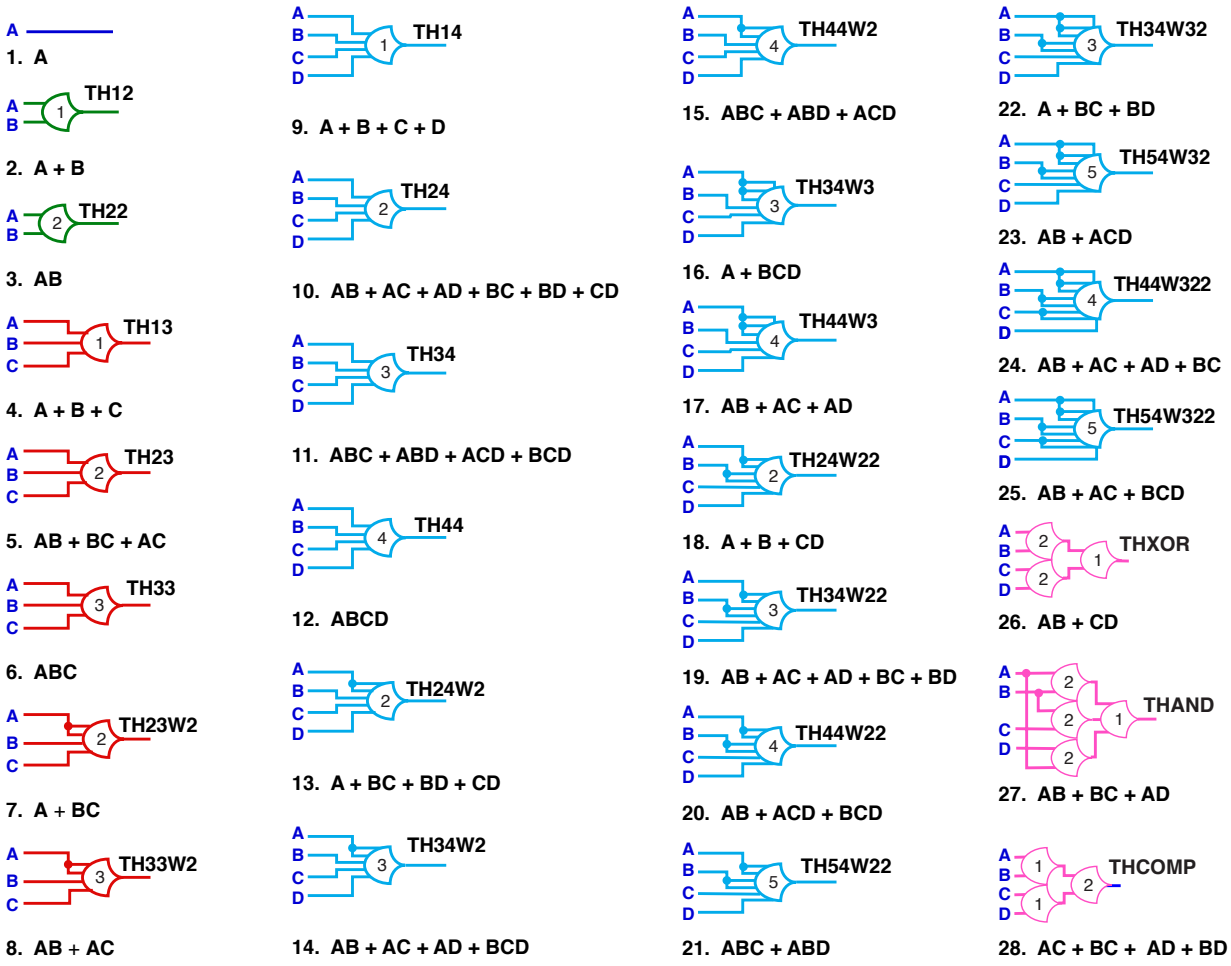


Figure A.2. The 28 NCL library operators with their corresponding positive Boolean functions.

A.2. The strategic error

Relating so directly to Boolean logic turned out to not be as useful as anticipated. While we could specify an NCL network in terms of Boolean operations the correspondence was not direct. There were a lot of considerations beyond the Boolean equations involved. Also we could not just take clocked Boolean designs and map them to NCL. The two logics behaved very differently.

Furthermore, relating to Boolean logic implied that Boolean logic was somehow more fundamental than NCL. We were trying to establish NCL as a superior form of logic to a population who by training and experience accepted Boolean logic as conceptually fundamental, a minimal necessary and sufficient theoretical standard to which all else reduces. Relating to Boolean logic just reinforced this view. It took many years of experience to realize that relating NCL to Boolean logic was more confusing then clarifying.

While everyone understood Boolean logic, threshold logic was not a current topic of familiarity. Its time had passed without it becoming used or widely understood. We were relating to a dinosaur, an elegant dinosaur but a dinosaur nonetheless.

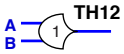
A.3. The conceptual error

The logic portrayed by the library of NCL operators is involved and complicated. For all its sense of completeness and closure with the 28 operators the library did not convey a sense of primitivity. Each operator had an assigned threshold and many of the inputs had assigned weights. Networks were difficult to design and optimize because each input had to be connected to a correctly weighted operator input.

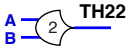
The conceptual error was to characterize this complicated set of behaviors as primitive. They seemed primitive. All but three operators were single gate threshold functions. I was aware that the gates had corresponding 1 of M and M of M networks shown in Figure A.3 but the threshold gates seemed like a higher level of primitivity, of atomicity and of efficiency. Even though there was a corresponding network of simpler operations the threshold operator was still conceptually a single gate, not a network, implying primitivity. The fact that it was a logic of threshold functions further implied an established primitivity.

Another confusion was that the operator hysteresis behavior was established with a single feedback signal further implying atomicity and primitivity.

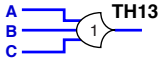
1. A



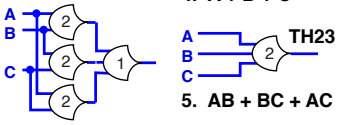
2. A + B



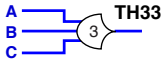
3. AB



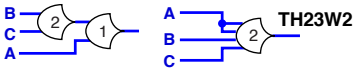
4. A + B + C



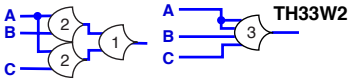
5. AB + BC + AC



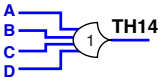
6. ABC



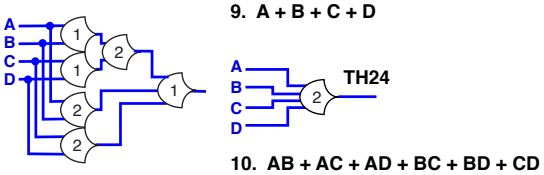
7. A + BC



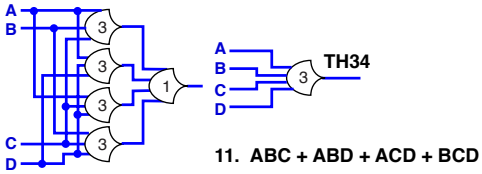
8. AB + AC



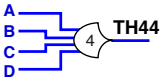
9. A + B + C + D



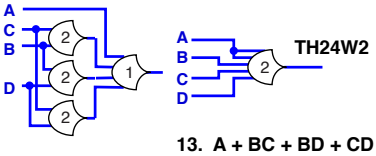
10. AB + AC + AD + BC + BD + CD



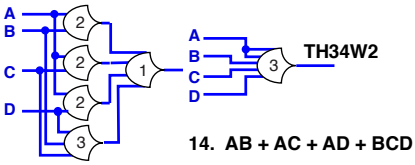
11. ABC + ABD + ACD + BCD



12. ABCD



13. A + BC + BD + CD



14. AB + AC + AD + BCD

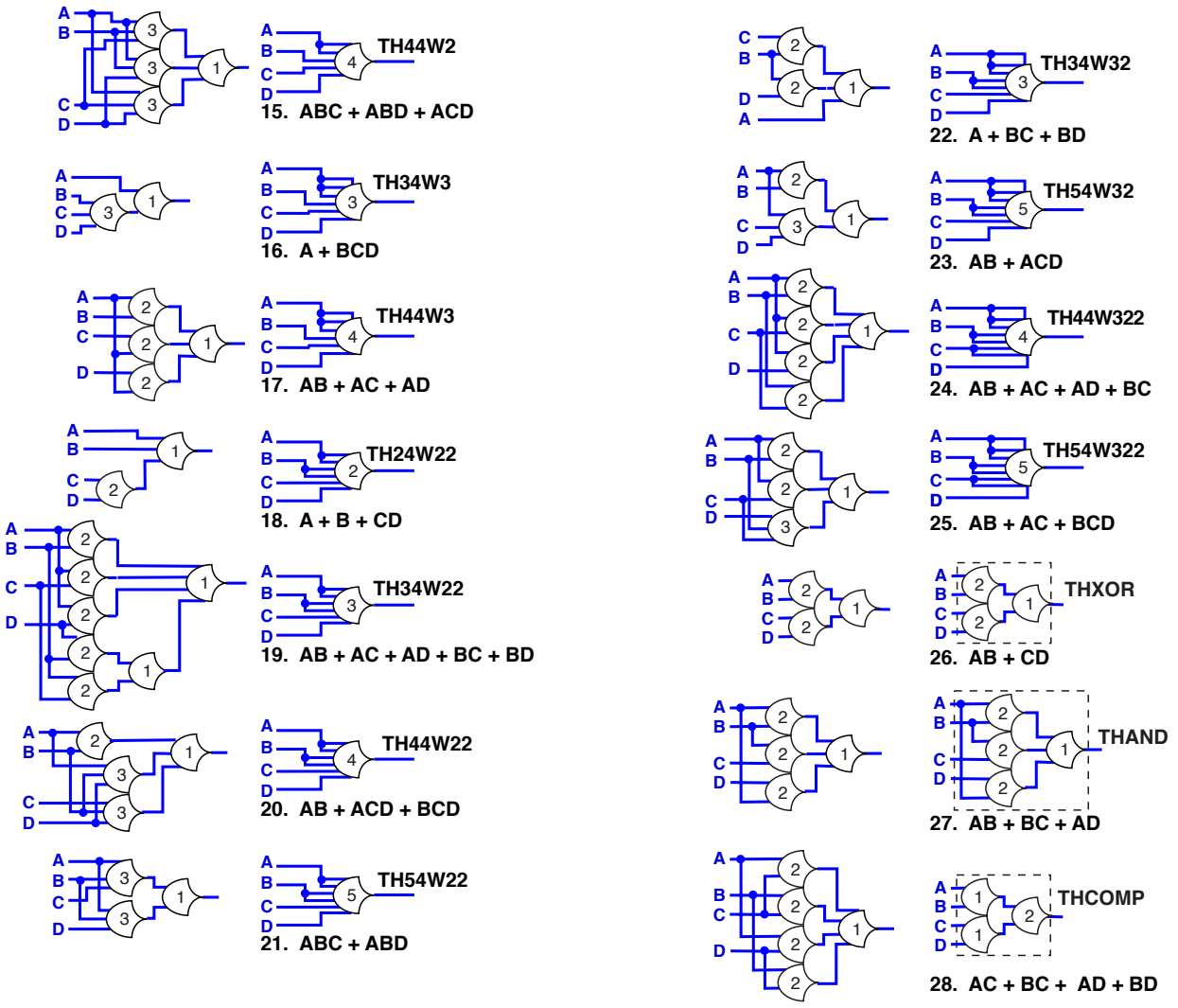


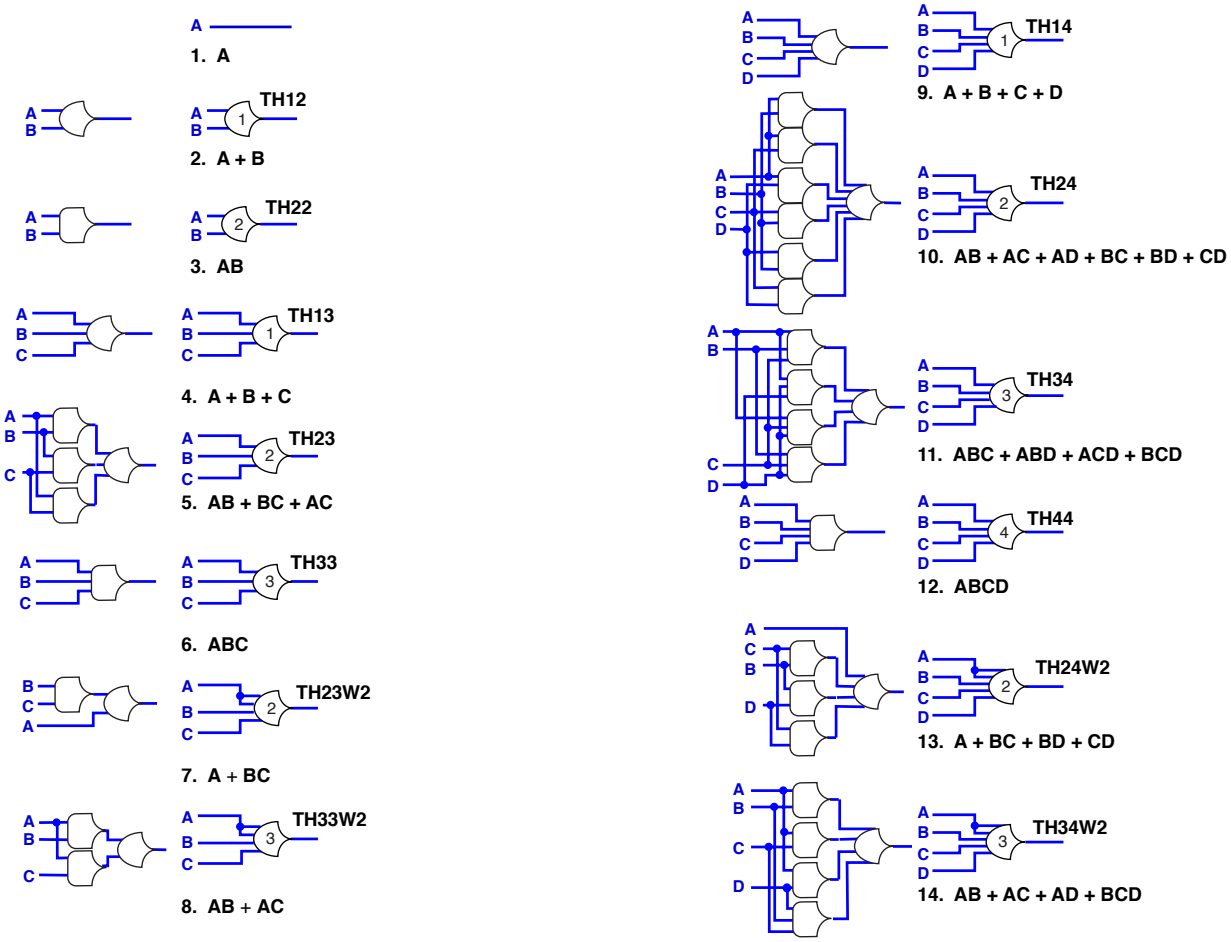
Figure A.3. NCL library with equivalent 1ofM and MofM networks.

The library seemed to embody a superior expressional efficiency and an elegant coherence or completeness. It was not until I started thinking in terms of "all of" and "one of" as primitives that I was able to stand outside the library and detach from Dave's and my creation.

A.4. The practical error

After studying many ways to implement the library gates including several implementations with actual threshold behavior it was finally decided to implement the gates with transistor switching networks. The decision was driven by what was possible and what was available. While there were fabrication techniques for threshold oriented devices using capacitors and other devices they were not as developed, as available or as efficient as the transistors. It was not so much an error as a constraint of practicality. So the elegant threshold behaviors were implemented with Boolean switching networks which, as it turned out, did not compete well with clocked Boolean switching networks.

Other possibilities of implementing the behaviors with inherent threshold and hysteresis behavior such as quantum dots, magnetic domains, memristors for instance may become practical in the future.



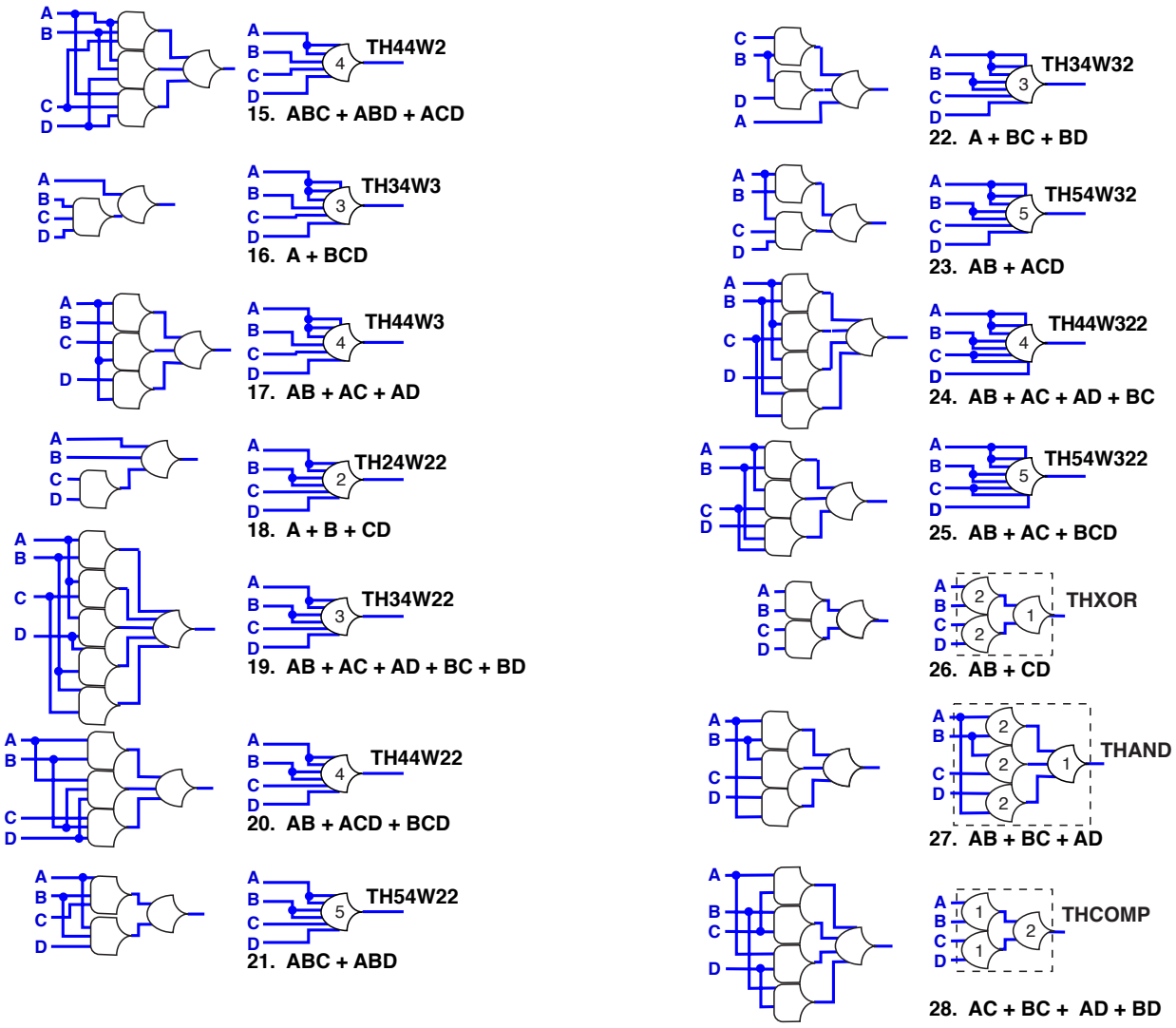


Figure A.4. The NCL library in terms of "one of" and "all of" behaviors.

A.5. The technical error

The technical error is the completeness behavior of the library operators which can lead to deadlock. In viewing the library operators as threshold functions, including the three operators that actually encapsulated threshold networks, they were considered as primitive with atomic behavior. In particular, completeness relations would apply across all inputs of each operator. When a particular quantity of inputs transitioned to "diff" an operator would transition its output to "diff" and when "all of" the inputs transitioned to "not diff" the operator would transition its output to "not diff". This completeness protocol worked in most circumstances but not always and the occasional failure is the fatal flaw. The difficulty is illustrated with the NCL THXOR operator in relation to its corresponding network.

A.5.1. The nominal behavior of THXOR

Figure A.5 shows the behavior of THXOR (top) and its associated network (bottom) is valid if [A,B] and [C,D] trace back to a "one of" relationships and are mutually exclusively presented.

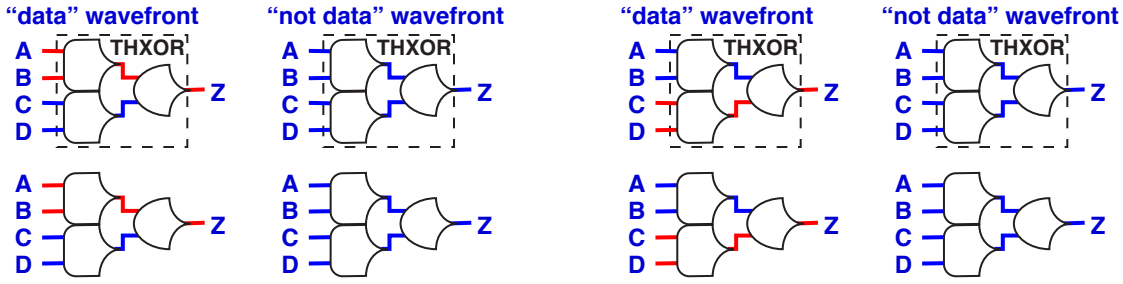


Figure A.5. Nominal behavior of THXOR and its associated network

A.5.2. The race behavior of THXOR

The inputs of every “one of” behavior must be mutually exclusive, only one transition to “diff” per “diff” wavefront. If [A,B] and [C,D] are not “one of” related they can collide and race to the “one of” behavior.

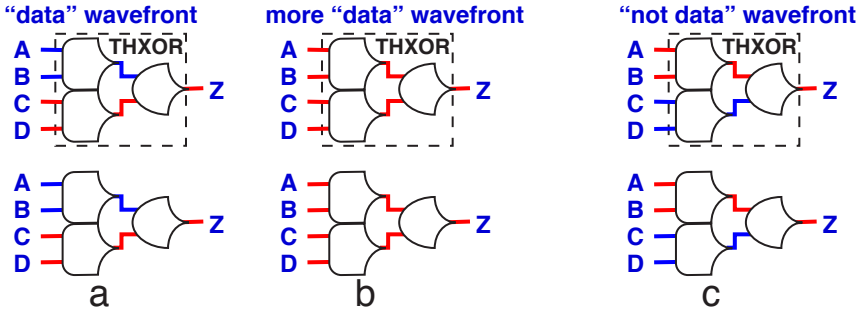


Figure A.6. [A,B] and [C,D] wavefronts collide.

In Figure A.6.a wavefront [C,D] transitions to “diff” first and Z transitions to “diff”. In Figure A.6.b wavefront [A,B] transitions to “diff” while [C,D] is still at “diff” but Z is already transitioned to “diff”. In Figure A.6.c the [C,D] “not diff” wavefront arrives. Z does not transition to “not diff”. [A,B] will never close and the network is deadlocked. The not coordinated wavefronts can also glitch Z which is equally disastrous.

A.5.3. The deadlock behavior of THXOR

Ensuring that any “one of” combination behavior has only one input transition to “diff” at a time is a basic design rule. The direct means of ensuring that the [A,B] and [C,D] wavefronts are mutually exclusive is for one component of each wavefront to come from the same 1composite locality. For instance, in Figure A.7 A and C are replaced with S/0 and S/1 from

locality S/{1,0};

S/0 will enable B. S/1 will enable D. This is correct in that there will be only one input to the “one of” behavior at a time. The deadlock difficulty with THXOR arises if either B or D transition to “diff” during the “diff” phase of the other. Figure A.7 illustrates D transitioning while [S/0,B] is “diff”.

In Figure A.7.a S/0 enables B and Z transitions to “diff”. In Figure A.7.b D transitions to “diff” while [S/0,B] are still at “diff”. In Figure A.7.c [S/0,B] transitions to “not diff” but the THXOR output Z does not transition to “not diff” because the input to THXOR is not completely “not diff”. The protocol for [S/0,B] is not completed and never will be. D is waiting on S/1 which will never occur because S/0 will never be closed and released. THXOR is deadlocked. If B and

D transition to “diff” mutually exclusively, which is typical, then the THXOR operator works just fine.

In the corresponding network of Figure A.7.c the completeness is across the individual combination behaviors. D is not preventing the transition of Z so S/0 can complete its oscillation and S/1 can eventually occur. Locality S is sufficient to ensure mutual exclusivity into the “one of” behavior and does not risk deadlock or race.

The network works correctly and the atomic threshold THXOR operator does not.

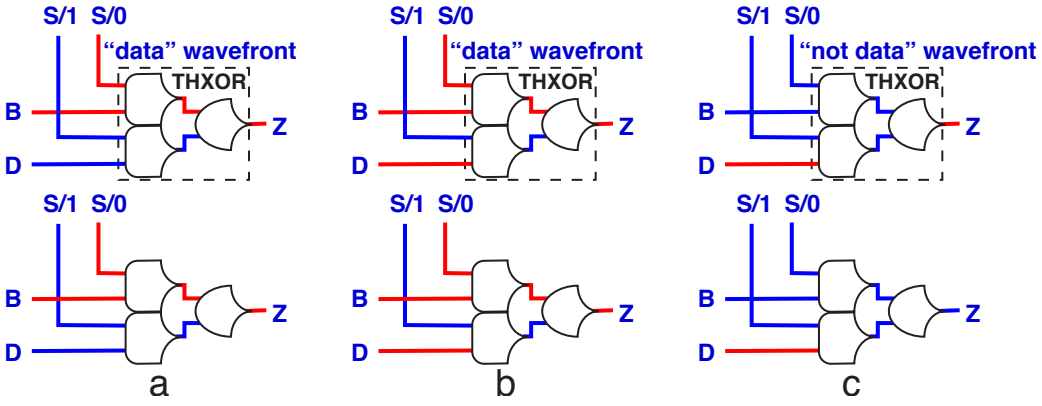


Figure A.7. The MUX protocol and deadlock.

The technical error is in treating the NCL operations as atomic primitives with completeness across all the inputs. This deadlock possibility is present with any NCL threshold operator that includes “one of” relations.

The THXOR difficulty was pointed out to me by others particularly by my colleague Stephen Johnson on numerous occasions. I persisted in viewing the difficulty not as a flaw in the library but as a problem of proper design rules. There is a necessary cooperation between the flow path behavior and the NCL operators. The paths have to monotonically transition and certain mutual exclusivities must be observed. My attitude was that this was just one of the mutual exclusivities that had to be observed. The fact that it rarely occurs contributes to perceiving it as a network design issue rather than a primitive atomic behavior issue. For many years I did not design complex circuits myself but relied on teams of engineers. It was only much later after I lost access to engineering support that I started designing significant NCL circuits myself. Only after I encountered several deadlock situations in my own designs and ended up backing out several threshold gates into networks of simpler gates to solve the completeness issue instead of altering the flow path behavior did I realize that the notion of atomic completeness across all inputs of many of the NCL operators is technically erroneous and that the characterization of the logic as threshold functions was misguided.

The difficulty follows from the dual threshold nature of the NCL operators. Traditional threshold logic is treated as passive logic functions that are coordinated with a clock. There was only one threshold that was either met or not at the clock tic. The next clock tic tested the threshold again. There was no behavior requirement imposed on the threshold function that reflected to the behavior of its input. However the input behaved the threshold gate just monitored the threshold.

With NCL there are two thresholds that have to be alternately matched so the input to the operator must behave properly by monotonically transitioning between completeness conditions. It is this behavior dependence between operator behavior and flow behavior that can be correct

by construction. But this cooperative behavior can also be incorrect by construction as is the case with the NCL operator completeness behavior. The criteria of correct construction can be subtler than expected.

1. Karl M. Fant, Logically Determined Design: Clockless System Design with NULL Convention Logic, (Hoboken, New Jersey, Wiley Interscience, 2005)

Appendix B: Pipeline Performance

B.1. Basic pipeline network structure and behavior

The first parameter of pipeline performance is the **oscillation network period**. The **oscillation path**, shown in Figure B.1, is characterized in two parts, the **wavefront flow path of the oscillation**, in green, and the **bubble flow path of the oscillation**, in orange*. The bubble flow path contains the oscillation closure flow path (section 3.7). The notion of bubble and bubble path will develop over the section. The combined delay of the two paths gives the **oscillation network period**. By convention, the wavefront path includes the enable rank of the input link and the bubble path includes the enable rank of the output link.

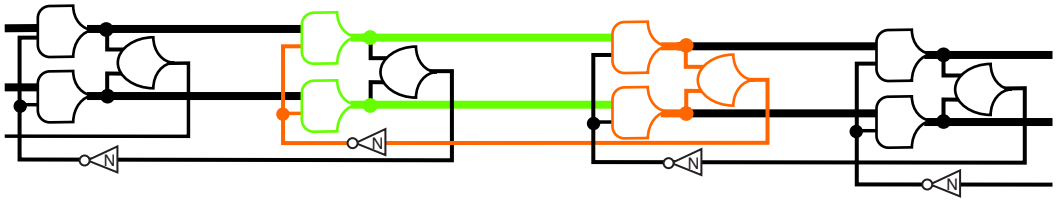


Figure B.1. Oscillation network oscillation/flow path.

Catenating the wavefront flow paths of the linked oscillations, Figure B.2, gives the **wavefront flow path of the pipeline** and adding the delays of the wavefront flow paths gives the **wavefront path delay** or **forward latency** of the pipeline.

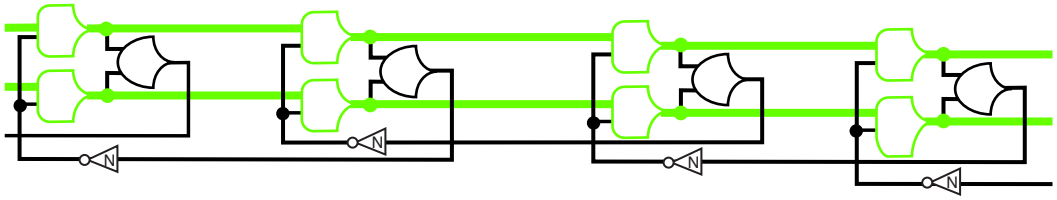


Figure B.2. Wavefront flow path, the forward latency of pipeline.

Catenating the bubble flow paths of the linked oscillations, Figure B.3, gives the **bubble flow path of the pipeline** and adding the delays of the bubble flow paths gives the **bubble path delay** or **reverse latency** of the pipeline.

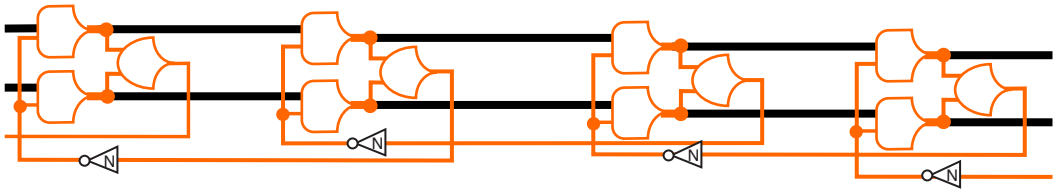


Figure B.3. Bubble flow path, the reverse latency of pipeline.

B.2. Primitive component performance

While delays in relation to some external time referent have no relevance to the correct behavior of a network, its performance in relation to an external time referent might be important to the network relating to behaviors external to the network. Its performance in relation to an

*. Bubble, an unusually apt asynchronous design term, is an emptiness into which wavefronts can flow.

external time referent can be determined in terms of subnetwork delays in relation to the external time referent.

Composing two half oscillations makes a complete oscillation. Each half oscillation contributes a portion of delay to each path of the complete oscillation. The half oscillation wavefront path delays are added to get the wavefront path delay for the new oscillation. The half oscillation bubble path delays are added to get the bubble path delay for the new oscillation for the new oscillation. Adding the wavefront path delay and the bubble path delay gives the period of the new oscillation.

Figure B.4 shows the input half oscillation and the output half oscillation of the quad to dual network component and the delays associated with the wavefront path and bubble path of each half oscillation. To keep the delay discussion simple (connections) have no delay and primitive search behaviors have delay based on their inputs.

1 input 20 ps
2 inputs 30 ps
3 inputs 40 ps
4 inputs 50 ps

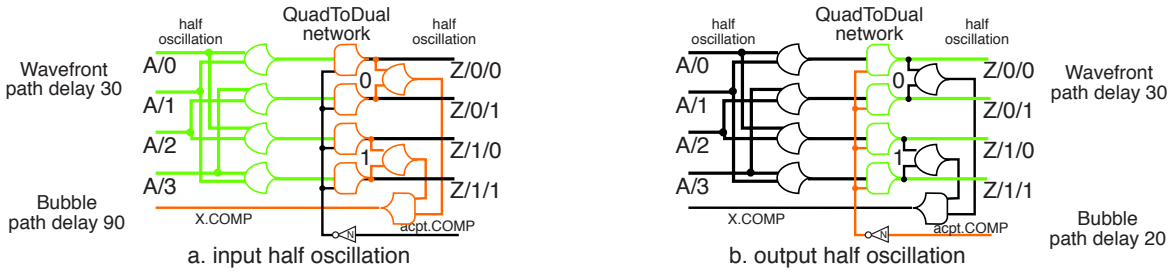


Figure B.4. Half oscillations of quad to dual primitive network.

Figure B.5 shows the delays associated with the half oscillations of the dual to quad network component.

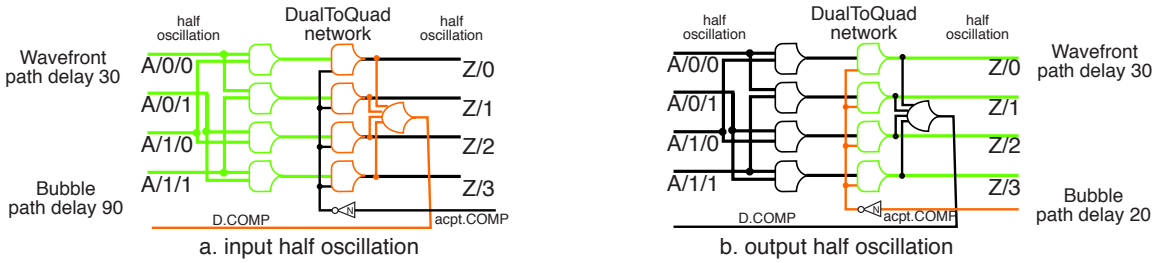


Figure B.5. Half oscillations of dual to quad primitive network.

B.3. Network composition

Networks are composed by associating the locality name of an output half oscillation to the corresponding locality name of an input half oscillation forming a full oscillation. The performance of the full oscillation is the combination of the performance of the half oscillations.

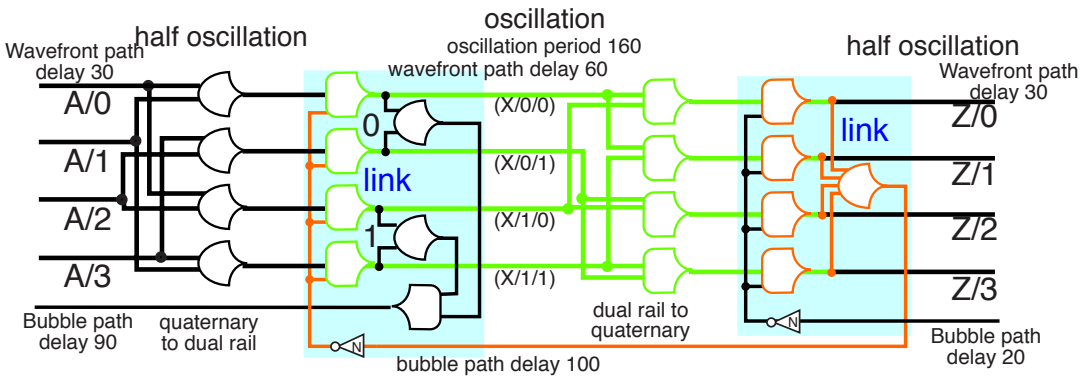


Figure B.6. The composed components form a pipeline network with one full oscillation.

```
network conversion (A -> Z):
locality (A, Z)/{3,2,1,0}/;
locality X/[1,0]/{1,0}/;
  QuadToDual (A -> X)
  DualToQuad (X -> Z)
endnetwork
```

B.3.1. Composition performance

The key factor of pipeline performance is the oscillation period. The performance of a boundary half oscillation is deferred until it is composed. For the composed network of Figure B.6 the oscillation period of the composed oscillation in network conversion is 160 ps so the maximum throughput of the pipeline is one “diff” wavefront every 320 ps. The pipeline will sustain any throughput presented to its input slower than or equal to 320 ps per “diff” wavefront. The initialization delay, the sum of the wavefront propagation delays, for network conversion is 120 ps. The initialization delay for the composed oscillation is 160 ps which in this case determines the initialization delay 160 ps plus margin, for the network. Usually the initialization delay is determined by the network wavefront propagation delay which is typically far longer than any individual oscillation period.

B.3.2. Further composition

Since the output and the input boundaries of network conversion are compatible half oscillations, two conversion networks can be connected forming a larger longpipe network. Both component networks propagate initialization so the composed network propagates initialization. Both networks propagate completeness behavior so the composed network accepts and propagates completeness behavior. All orphans remain isolated. The composed network, validated and bounded by half oscillations, is ready for further composition.

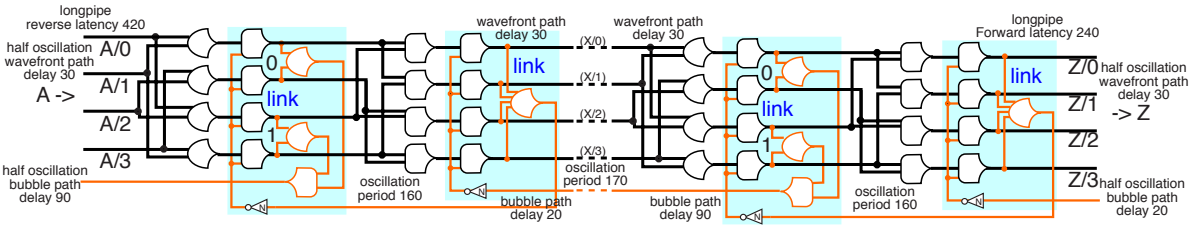


Figure B.7. Bigger network from smaller networks.

```
network longpipe (A -> Z):
locality (X, A, Z)/{3,2,1,0}/;
  conversion (A -> X)
```



```
  conversion (X -> Z)
endnetwork
```

B.3.3. Longpipe performance

Performance parameters are accumulated as composition progresses. In Figure B.7 the longest oscillation period is the new formed oscillation at 170 ps. The network will sustain a throughput of 340 ps per “diff” wavefront. The forward latency is the sum of the wavefront path delays, 240 ps. The reverse latency is the sum of the bubble path delays, 420 ps. The wavefront path initialization delay is 240 ps. The slowest oscillation initialization delay is 170 ps So the initialization delay for the new network is 240 ps plus margin.

B.3.4. Pipeline performance

The input and output of a pipeline are half oscillations which, for purposes of performance analysis, are assumed to be sufficient flow in that when the input closure enables an input wavefront a wavefront is present and flows into the pipeline and when an output wavefront is presented its closure is present and the wavefront flows out of the pipeline. A pipeline network is a unit of composition that flows with a characteristic throughput which equals the period of its slowest oscillation and with a characteristic latency which is the delay of its wavefront path.

B.3.5. Performance in an environment

If the environment responds more slowly than the pipeline’s characteristic throughput then the pipeline will conform to the throughput of the environment. If the environment can conform to the throughput of the pipeline then the system will flow at the characteristic throughput rate of the pipeline. If the environment demands a faster throughput than the pipeline can support then the environment/pipeline system will fail.

Appendix C: Wavefront Arbitration

Uncoordinated wavefront flows that share a common resource must be arbitrated into mutually exclusive coordinated flow behavior with only one wavefront proceeding at a time. Wavefronts flow in “not diff”/“diff” wavefront pairs. An arbiter behavior will allow only one wavefront pair to flow at a time.

C.1. The MUTEX

The mutual exclusion behavior of an arbiter is provided by a MUTEX. Figure C.1 Shows a CMOS implementation of a MUTEX, an SR flipflop with a filter across the two outputs to minimize metastable oscillation and facilitate resolution of the metastability.

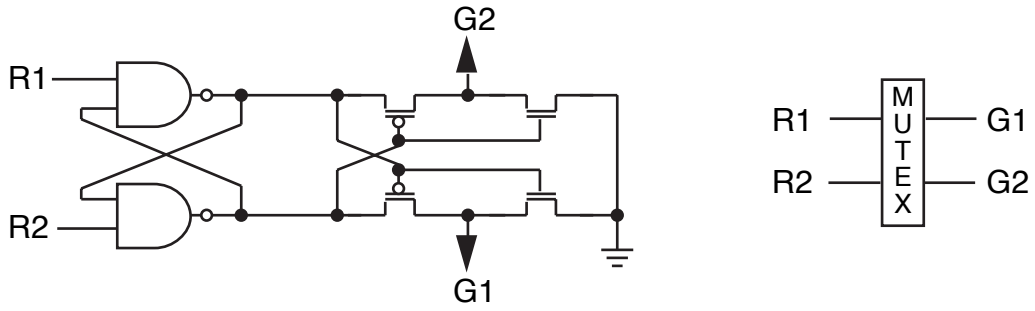


Figure C.1.MUTEX circuit and symbol.

A MUTEX is a primitive mutual exclusion behavior. that will assert grants G1 or G2 mutually exclusively in time. They are asserted, respectively, in response to the requests R1 or R2 which can be asserted at any time. If asserted simultaneously the MUTEX will chose one request to grant and block the other request until the granted request is released. When the granted request is released its grant will be removed and the waiting request will be immediately granted. If one request arrives first it is granted and the second arriving request waits until the first request is released upon which the waiting request is immediately granted. If a request is released before it is granted then it is not granted and has no effect. So there is an assumption that the behaviors asserting R1 and R2 are sensitive to their grants, will wait appropriately and that the assertion of the grant will ultimately cause the release of the request.

C.2. The arbiter

A wavefront arbiter must allow a “not diff”/“diff” wavefront pair. A MUTEX request occurs with the arrival of a “diff” wavefront and the transition of request to “diff”. With the arrival of the “not diff” wavefront the request will transition to “not diff” and the MUTEX will immediately allow a waiting grant. This does not allow for the passage and completeness of the “not diff” wavefront before allowing the next grant. There must be an additional mechanism in addition to the MUTEX that waits for the “not diff” wavefront to completely propagate before granting the waiting request. Figure C.2 shows the arbiter core with the extra circuitry to wait on the “not diff” wavefront.[1] The arbiter core fits into an oscillation and arbitrates wavefront flow through the oscillation.

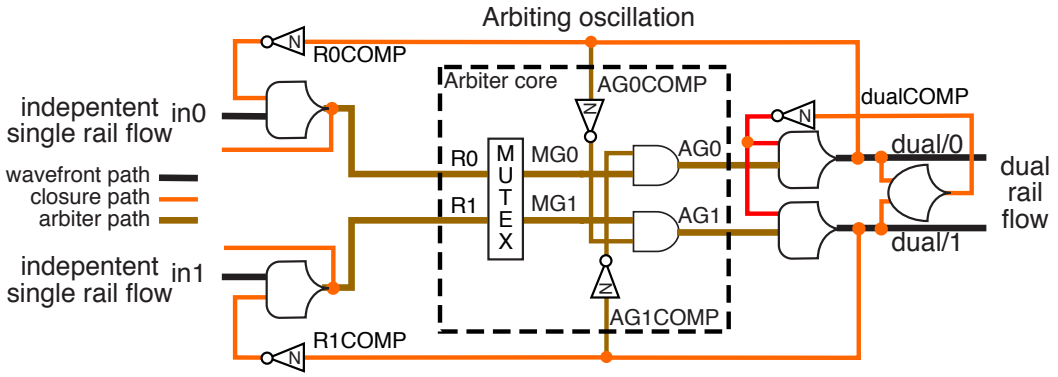


Figure C.2.Arbitrating oscillation.

In the following example an arbitrating oscillation receives two uncoordinated single rail flows and produces a single coordinated dual rail flow. In Figure C.3 the arbiter oscillation begins in an initial configuration awaiting wavefront flow.

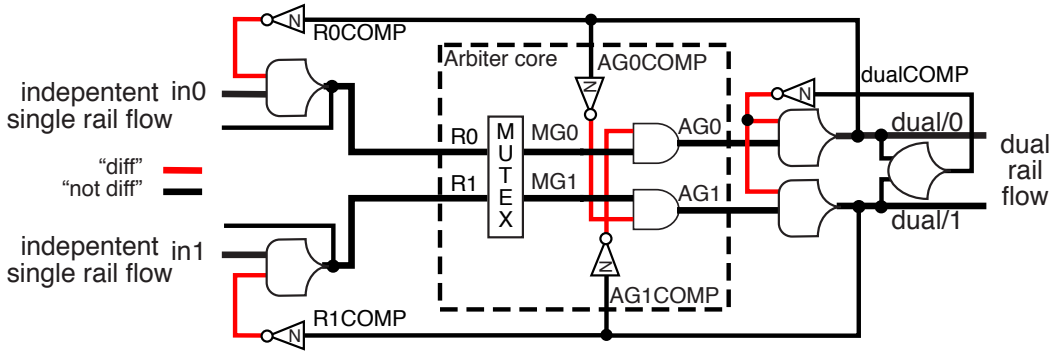


Figure C.3.Arbiter oscillation awaiting wavefront flow.

"diff" wavefronts arrive simultaneously at R0 and R1.

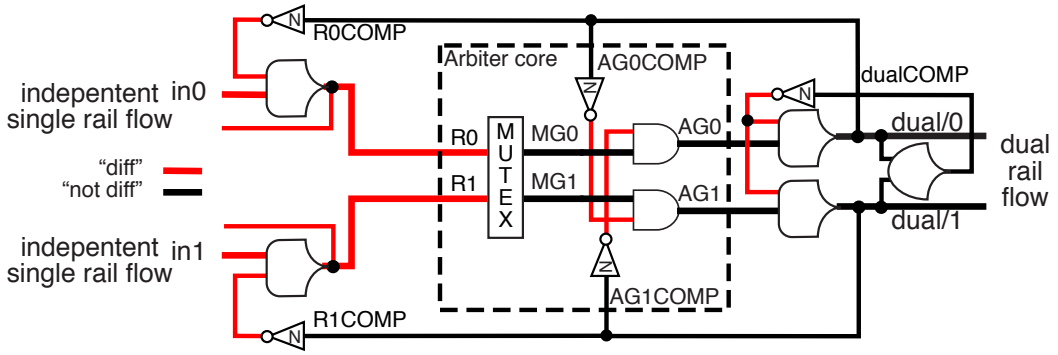


Figure C.4.Request "diff" wavefronts arrive simultaneously at MUTEX.

In Figure C.4 two "diff" wavefronts arrive simultaneously at the MUTEX.

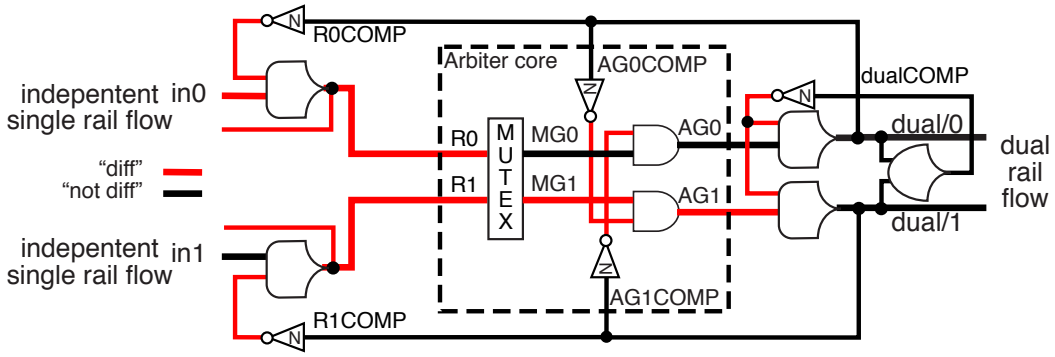


Figure C.5.The MUTEX grants R1 and blocks R0.

In Figure C.5 request R1 is granted with MUTEX Grant MG1 and Arbiter Grant AG1. R0 is blocked by the MUTEX.

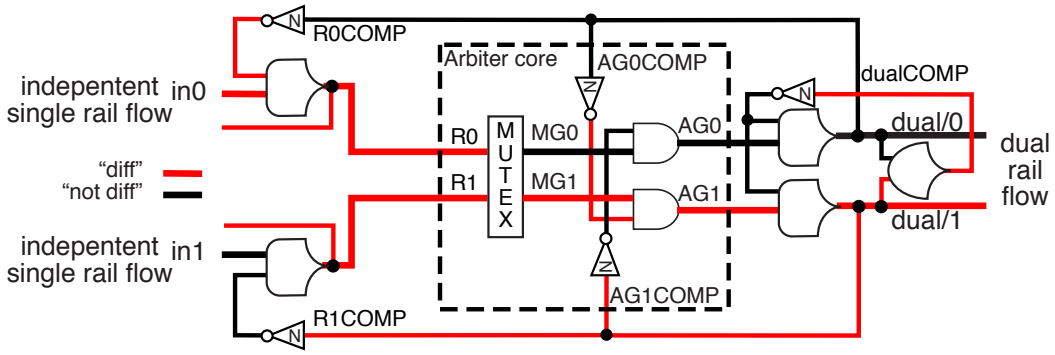


Figure C.6.The R1 flow closes with in1 enabling the "not diff" wavefront.

In Figure C.6 R1 flows through the link and closes with in1 enabling the "not diff" wavefront flow. The closure also closed with the arbiter core through AG1COMP disabling AG0.

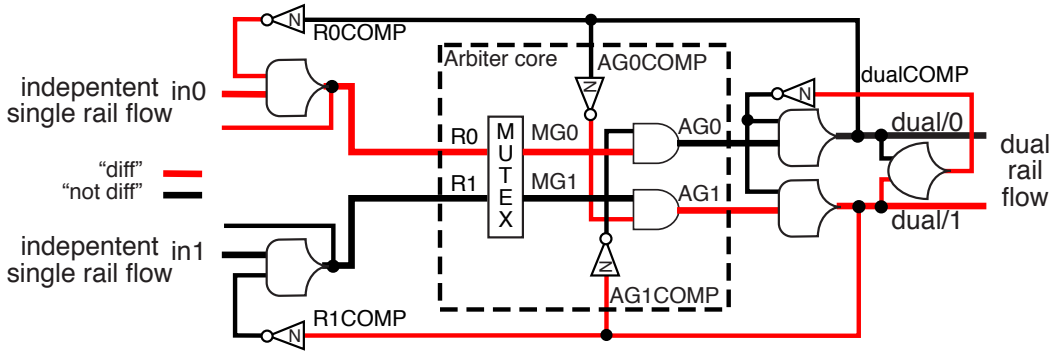


Figure C.7.R1 is released and R0 is immediately granted by the MUTEX.

In Figure C.7 the R1 "not diff" wavefront flows to the MUTEX releasing the R1 request. The MUTEX immediately grants the R0 request with MG0 which is blocked at AG0. **There is a local time relation here in that the AG1COMP flow must arrive at AG0 before the "not diff" wavefront arrives at R1 and releases the request.**

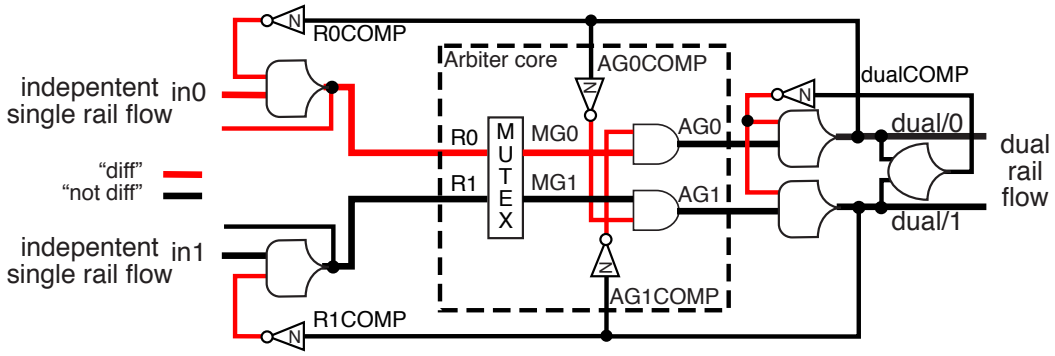


Figure C.8.The "not diff" wavefront closure released the block of AG0.

In Figure C.8 the R1 "not diff" wavefront flow closure flows through AG1COMP and releases the AG0 block.

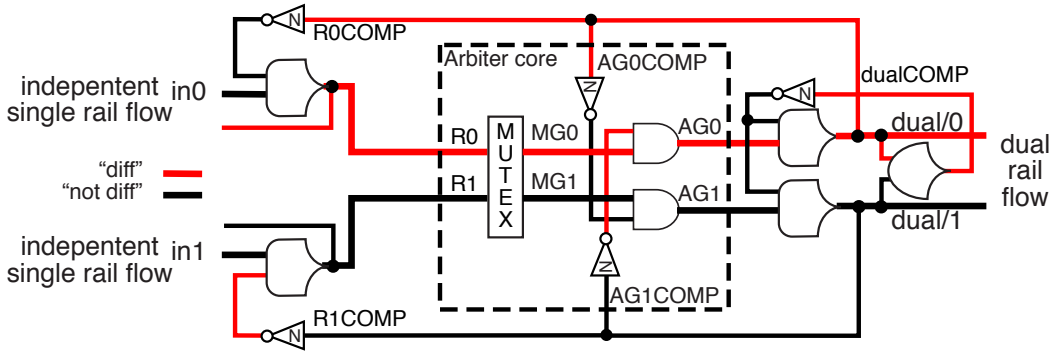


Figure C.9.The R0 "diff" wave flows.

In Figure C.9 the R0 "diff" wavefront flows through AG0 and the link. The "diff" closure flowing through AG0COMP blocks AG1 and enables the "not diff" wavefront at in0.

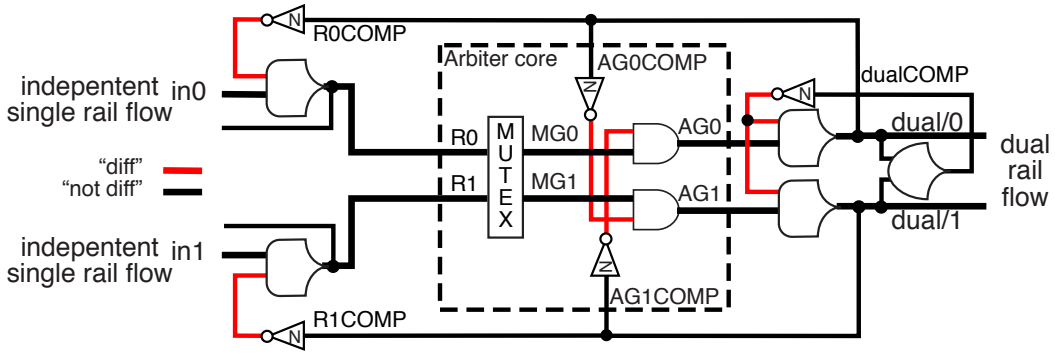


Figure C.10.The R0 "not diff" wavefront flows.

In Figure C.10 the R0 "not diff" wavefront flows through the link closing with in0, releasing AG1 and the arbiter oscillation returns to its initial configuration awaiting the next "diff" wavefront.

C.3. Arbitrating composite wavefronts

Figure C.11 shows an arbitrating oscillator arbitrating uncoordinated dual-rail flow paths through a "one of" behavior into a single dual-rail flow path. Large composite wavefronts can be

arbitrated. The completeness of the dual-rail wavefront is used as the request. The grant enables the flow of the wavefront through the grant link.

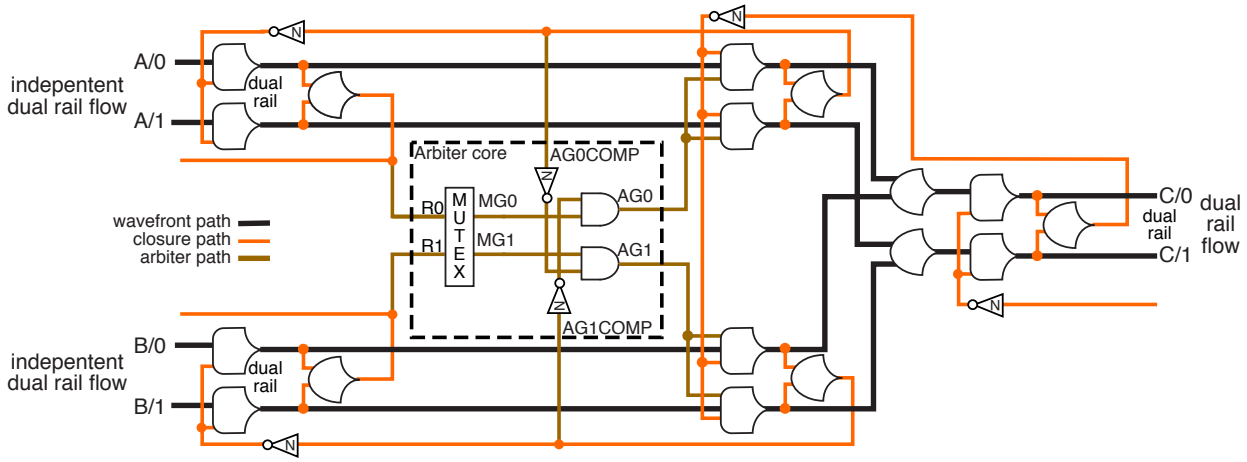


Figure C.11.Two dual rail paths arbitrated into “one of” behavior.

Figure C.12 shows large composite independent wavefronts arbitrated into a single composite wavefront flow. Any one rail portion of the completeness of a wavefront can be used as the request. The grant is used to enable the flow of the wavefront through the grant link. The completeness of flow is managed between the grant link and the requesting link. The completeness of the grant link is reduces to a single rail closure which presents to AGxCOMP and closes with the requesting link.

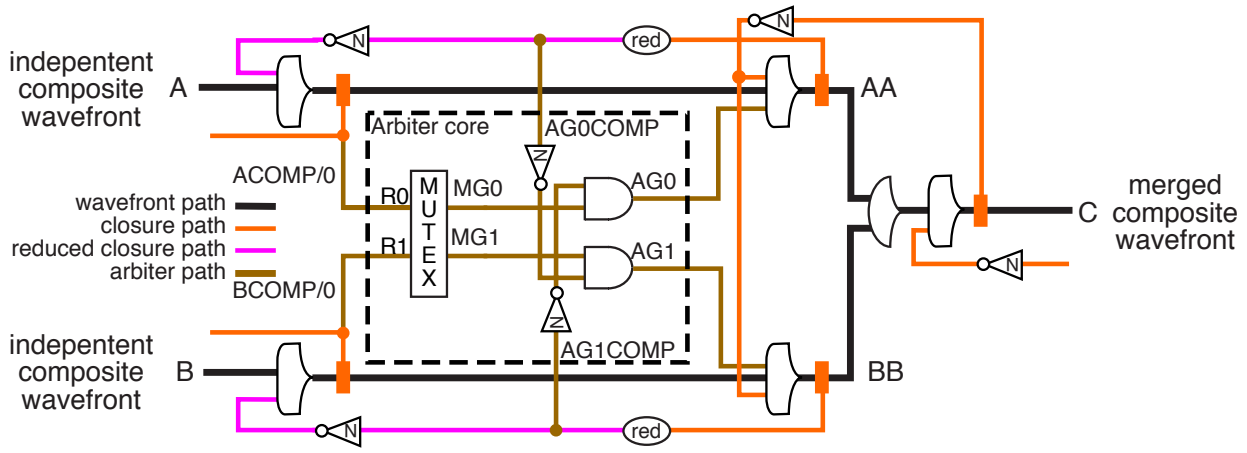


Figure C.12.Composite merge.

1. The arbiter core is derived from a standard asynchronous circuit called a tree arbiter the development of which is recounted in Teresa Meng, *Synchronization Design for Digital Systems, (Norwell, Massachusetts, Kluwer Academic Publishers, 1991). pp 158-163.*

Appendix D: Wavefronts and bubbles

Wavefronts flow forward into bubbles. Bubbles flow backward around wavefronts. This difficult to intuit counterflow behavior of wavefronts and bubbles is the fundamental dynamic flow behavior of a network of linked oscillations. Wavefronts and bubbles interact at a link. The link symbol of Figure D.1 is used in the following discussion.

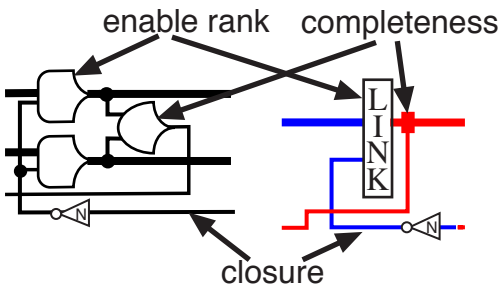
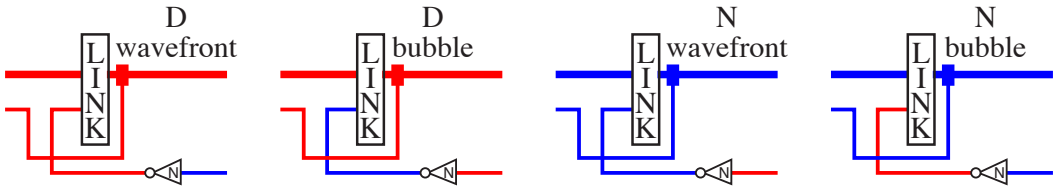
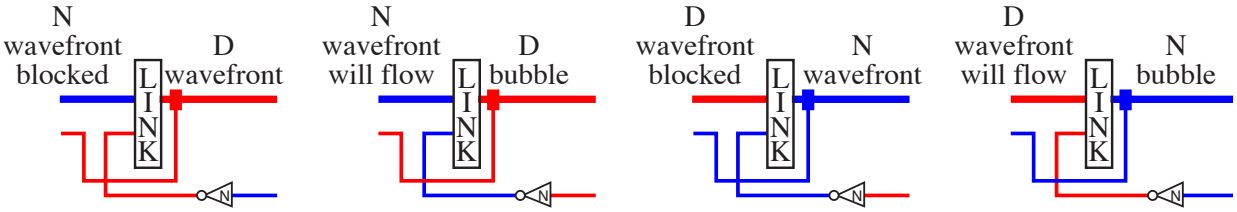


Figure D.1. The link symbol

A link presents a **D** wavefront when asserting **D** and its closure is enabling **D**.
A link presents a **D** bubble when asserting **D** and its closure is enabling **N**.
A link presents a **N** wavefront when asserting **N** and its closure is enabling **N**.
A link presents a **N** bubble when asserting **N** and its closure is enabling **D**.



A **D** wavefront cannot flow into a **N** wavefront.
A **D** wavefront can flow into a **N** bubble.
A **D** wavefront should never encounter another **D** wavefront.
A **N** wavefront cannot flow into a **D** wavefront.
A **N** wavefront can flow into a **D** bubble.
A **N** wavefront should never encounter another **N** wavefront.



D.1. Pipeline flow behavior

A **D** wavefront will flow into a **N** bubble. The **N** bubble flows around the **D** wavefront becoming a **D** bubble. A **N** wavefront will flow into a **D** bubble. The **D** bubble flows around the **N** wavefront becoming a **N** bubble. Another way to say it is that **D** Wavefronts flow through an empty background of **N** bubbles trailing **D** bubbles followed by **N** wavefronts flowing through **D** bubbles trailing **N** bubbles restoring the background empty condition of **N** bubbles through which a successor **D** wavefront can flow.

A pipeline network is initialized by an initialization signal to an empty background condition of **N** bubbles. The initialization signal must be held long enough for the initializing **N** wavefront to propagate

through the entire pipeline. The initialization signal is the only aspect of network design that is necessarily associated with an external time referent.

D.1.1. Pipeline initialization:

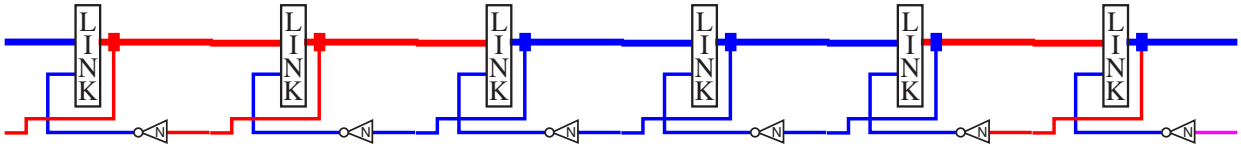


Figure D.2. Pipeline in random power up state with an initialization signal forcing the converters to assert N.

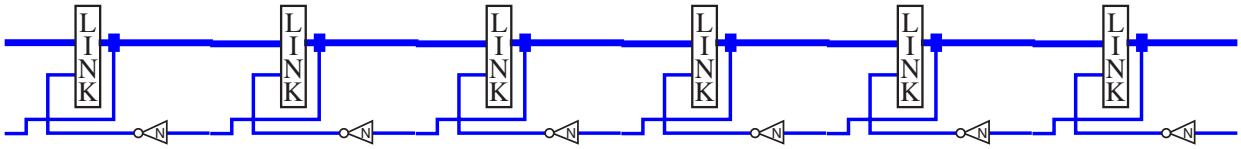


Figure D.3. A N wavefront presented at the input flows through entire pipeline.

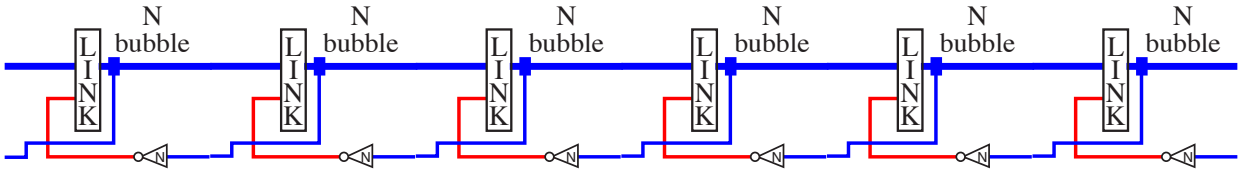


Figure D.4. Init is released and pipeline becomes filled with N bubbles waiting for a D wavefront.

D.1.2. Wavefront and bubble flow behavior

In Figure D.5 a **D** wavefront is presented to the input and flows into the **N** bubbles through the entire pipeline leaving behind **D** bubbles waiting for an **N** wavefront. It is not accepted at the output and becomes stalled at the output as a **D** wavefront, .

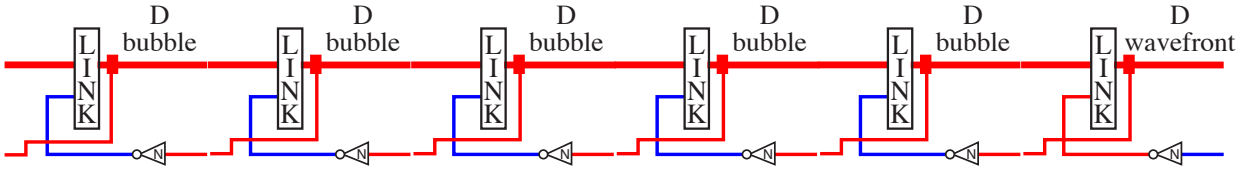


Figure D.5. A D wavefront flows through N bubbles.

In Figure D.6 a following **N** wavefront is presented to the input and flows into the **D** bubbles leaving behind **N** bubbles. It cannot flow into the **D** wavefront and is blocked, .

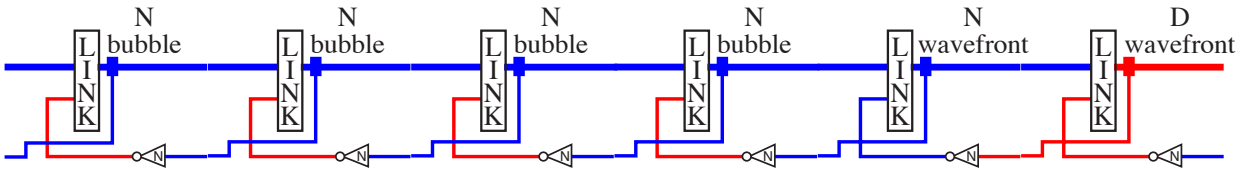


Figure D.6. A N wavefront flows through D bubbles.

In Figure D.7 successive wavefronts are presented to the input and flow into the pipeline being successively blocked and filling the pipeline with wavefronts waiting for the first **D** wavefront to be accepted.

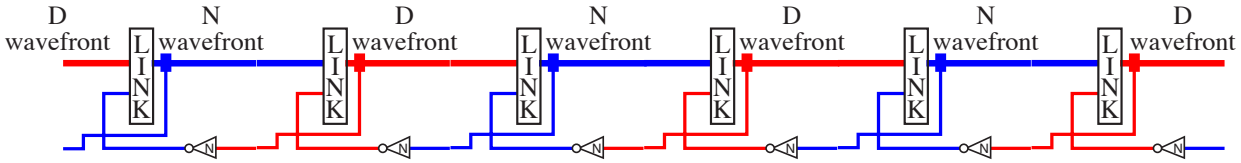


Figure D.7. Pipeline filled with blocked wavefronts.

In Figure D.8 the first **D** wavefront is accepted and a **D** bubble presents at the output of the pipeline.

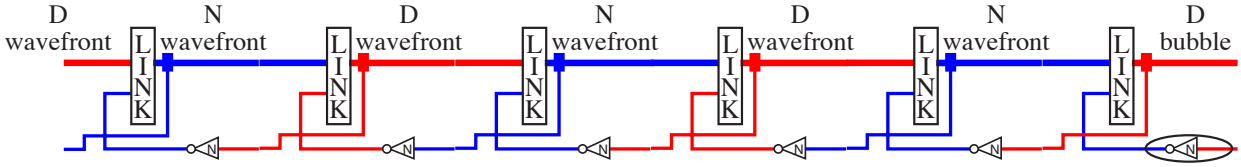


Figure D.8. Output D wavefront is accepted and D bubble enters pipeline.

In Figure D.9 the **N** wavefront flows into the **D** bubble leaving behind an **N** bubble.

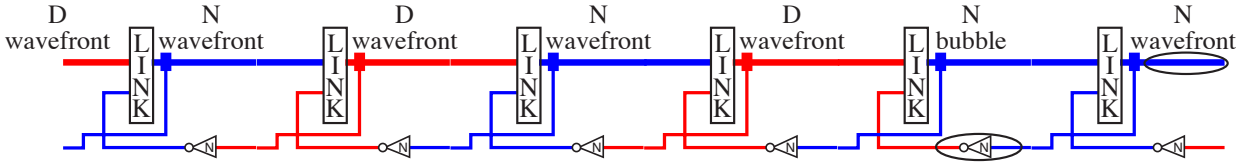


Figure D.9. N wavefront flows into D bubble.

In Figure D.10 the **D** wavefront flows into the **N** bubble leaving behind a **D** bubble.

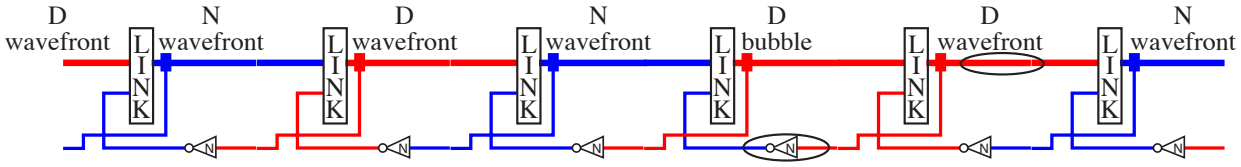


Figure D.10. D wavefront flows into N bubble.

In Figure D.11 the **N** wavefront flows into the **D** bubble leaving behind a **N** bubble. The rightmost **N** wavefront flows out of the pipeline leaving behind an **N** bubble.

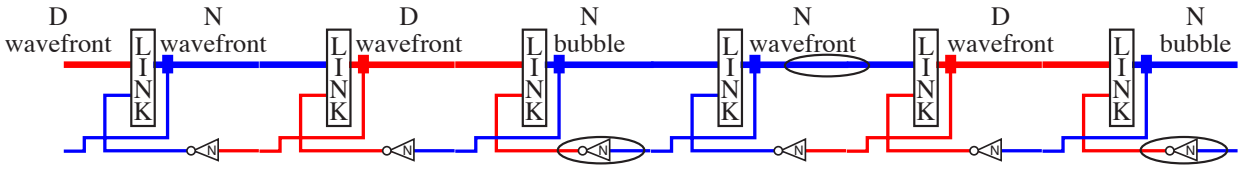


Figure D.11. N wavefront flows into D bubble.

In Figure D.12 the **D** wavefronts flows into both **N** bubbles leaving behind a **D** bubbles ... And so on.

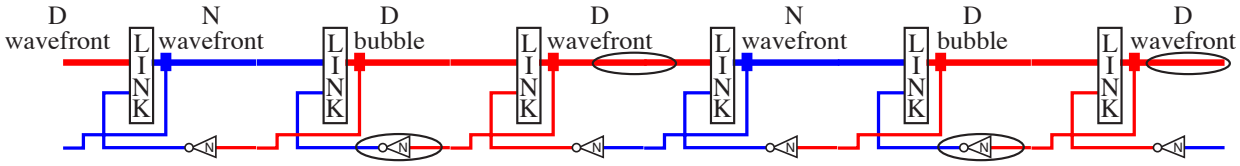


Figure D.12. D wavefront flows into N bubble.

The bubble, alternating between **N** bubble and **D** bubble, flows backward through the waiting wavefronts in the pipeline. Each wavefront progresses one oscillation with each bubble entering the pipeline and flowing through the wavefronts. As each wavefront flows forward into a bubble a bubble flows backward around the wavefront. In the absence of bubbles wavefronts do not flow. In the absence of wavefronts bubbles do not flow.

Appendix E: Initializing a D wavefront in a pipeline

All interaction networks need to be initialized at inception. A pipeline network of linked oscillations initializes with all conversions forced to **N** and with completely **N** presented to the network inputs which flows through all the links and through each oscillation network to initialize the entire pipeline network to **N** or empty. The init signal, which has to be held long enough for the **N** conditions presented to the input of the network to propagate through the entire network, is the only signal in an interaction network with a specific and extrinsic timing requirement. When the init signal is released, all conversions transition to **D** filling the network with **N** bubbles (Appendix D) enabling the first transition to **D** wavefront to flow into and through the network.

E.1. Initializing a D wavefront

Occasionally it is necessary to initialize a **D** wavefront in a pipeline network otherwise initialized to all **N**. An individual closure can be initialized to **D** or **N**. Initialization is specified with **D** or **N** in the primitive behavior graphic symbol which is also connected to the init signal. Initializing a **D** wavefront in a pipeline involves two consecutive oscillation networks as shown in Figure E.1. The initializing oscillation network initializes a **D** completeness wavefront across the closure flow path. The oscillation networks receiving the initialized wavefront must block each initialized **D** with an initialized **N** to present a **N** initialization wavefront to the rest of the pipeline. The closures of the two initializing oscillation networks use uninitialized converters. When the init signal is released the **D** wavefront begins spontaneously flowing through the pipeline.

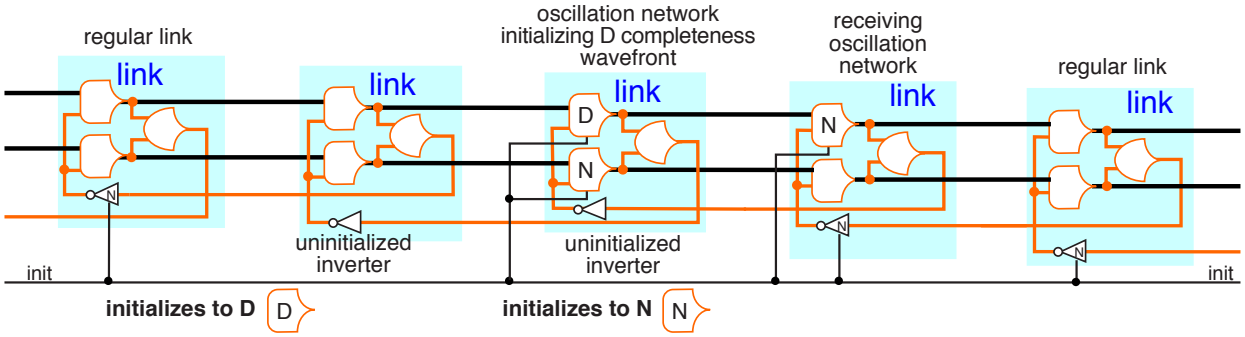


Figure E.1.Initializing a D wavefront in a pipeline network.

Cover the “allof” to D issue the shared closure

Appendix F: 5 bit Multiply interaction network

The **5bitmul** requires two component networks not yet defined **halfadd** and **mul**. The **halfadd** expression below specifies the **halfadd** interaction network in the left of Figure F.1.

bound reference name **binding portal localities**

(halfadd(A/{1 0}/ B/{1 0}/ -> sum/{1 0}/ Cout/{1 0}/)
sum/{1<={[A/0 B/1] [A/1 B/0]}
0<={[A/0 B/0] [A/1 B/1]} }
Cout/{1<=[A/1 B/1]
0<={[A/0 B/0] [A/0 B/1] [A/1 B/0]} })

dependency relations

sum is dependent on the dependency relations of its components, **{1 0}** on input localities **A** and **B**. **Cout** is dependent on the dependency relations of its components, **{1 0}** on input localities **A** and **B**

The **mul** expression below specifies the **mul** interaction network in the right of Figure F.1.

bound reference name **binding portal localities**

(mul(A/{1 0}/ B/{1 0}/ => P/{1 0}/)
P/{1<=[A1 B1]
0<={[A/0 B/0] [A1 B0] [A/0 B/1]} })

dependency relations

P is dependent on the dependency relations of its components, **{1 0}** on input locality **A** and **B**.

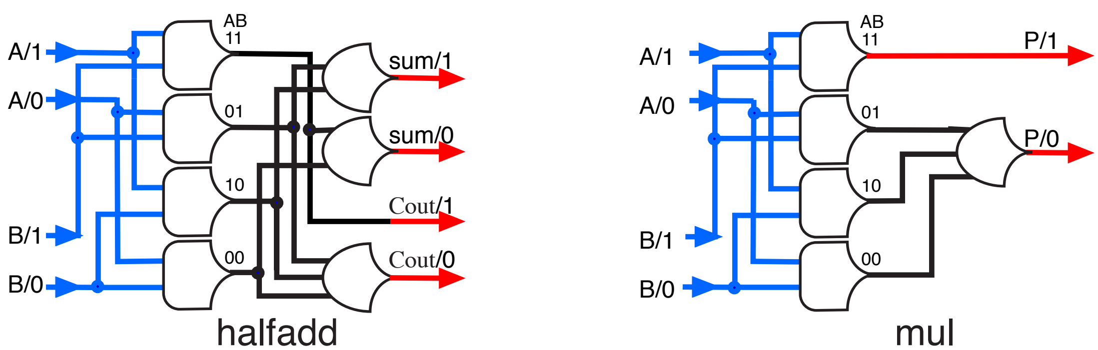


Figure F.1. halfadd and mul component networks

The **5bitmul** references component networks **fulladd**, **mul** and **halfadd** all of which are constant and fulfill the completeness criterion. The **5bitmul** expression below specifies the 5 bit multiply interaction network of Figure F.2.


```
bound reference name          binding portal localities
                  (5bitmul(A/[4-0]  B/[4-0] -> P/[9-0])
internal localities  ((C11 C12 C21 C31 C22 C13 C41 C32 C23 C14 C51 C42 C32
                       C23 C33 C24 C52 C43 C34 C53 C44 C53 C54)/{1 0}/)
                     P/[9<=C54
                        8<=fulladd(mul(A/4 B/4) C44 C53 =>  #  C/54)
                        7<=fulladd(fulladd(mul(A/4 B/3) C43 C52 =>  # C53) mul(A/3 B/4) C34 =>  #  C/44)
                        6<=fulladd(fulladd( fulladd(mul(A/4 B/2) C42 C51 =>  # C52)
                                   mul(A/3 B/3) C33 =>  # C43) mul(A/2 B/4) C24 =>  #  C/34)
                        5<=fulladd(fulladd( fulladd( halfadd( mul(A/4 B/1) C/41 =>  # C51)
                                   mul(A/3 B/2) C32 =>  # C42)
                                   mul(A/2 B/3) C23 =>  # C33) mul(A/1 B/4) C14 =>  #  C/24)
dependency relations    4<=halfadd(fulladd( fulladd( fulladd( mul(A/4 B/0)  mul(A/3 B/1) C/31 =>  # C41)
                                   mul(A/2 B/2) C22 =>  # C32)
                                   mul(A/1 B/3) C13 =>  # C23) mul(A/0 B/4) =>  #  C/14)
                        3<=halfadd(fulladd( fulladd( mul(A/3 B/0)  mul(A/2 B/1) C/21 =>  # C31)
                                    mul(A/1 B/2) C12 =>  # C22) mul(A/0 B/3) =>  #  C/13)
                        2<=halfadd(fulladd( mul(A/2 B/0)  mul(A/1 B/1) C/11 =>  # C21) mul(A/0 B/2) =>  -  C12)
                        1<=halfadd(mul(A/1 B/0) mul(A/0 B/1) =>  #  C11)
                        0<=mul(A/0 B/0 )
                         ]   )
```

5bitmul inherits locality structure in the same way as the 5bitadd above.

(5bitmul(A[4-0]/{1 0}/ B[4-0]/{1 0}/ => P[9-0]/{1 0}/) (…) …)

P is dependent on the dependency relations of its components, **[9-0]** on input localities **A** and **B** and the internal **Cxx** localities. The detailed relations are too involved to verbalize.

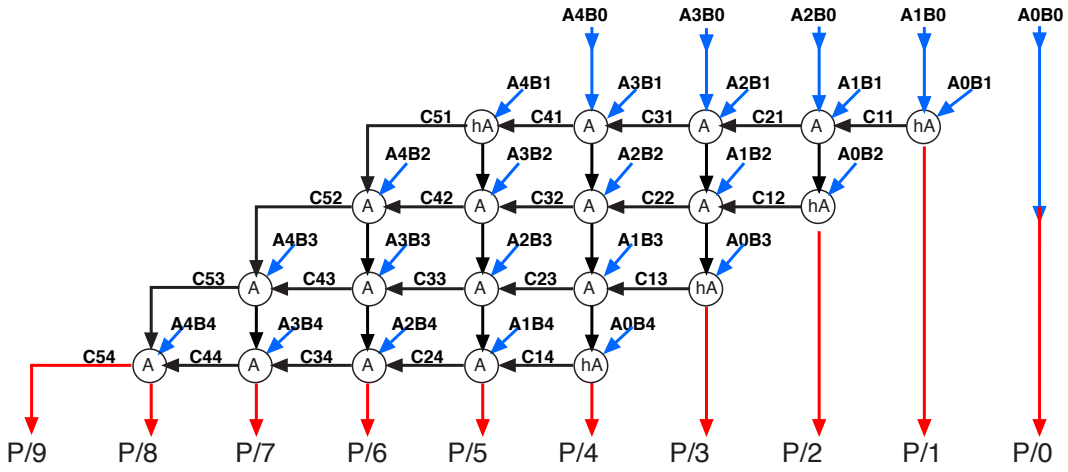


Figure F.2. 5 bit multiply interaction network.

Each **fulladd**, **halfadd** and **mul** component network in the **5bitmul** network is constant and fulfills the completeness criterion so **5btmul** as a whole is constant and fulfills the completeness criterion. If any part of **A** or **B** is not completely formed a component network input will not completely form, the component network will not transition its output and a part of **P** will not transition. If **P** transitions to completeness it means that the inputs **A** and **B** have transitioned to completeness.

Appendix G: The orphan delay risk

With the emergence of completeness closure and the oscillation network of section 3.7 the orphan risk can be characterized. The delay risk of the orphan is in the **N** wavefront not fully propagating before the next transition to **D** wavefront arrives at an orphan branch. The next transition to **D** wavefront cannot arrive at an orphan branch until the transition to **D** wavefront is allowed into the network by the closure. Consequently, successive **D** wavefronts cannot flow faster than the oscillation period of the network. The delay risk of the orphan can be evaluated by determining:

1. that all orphan branches are isolated to a single branch path within the interaction network and do not propagate through a primitive interaction behavior,
2. That the shortest orphan branch delay contributes to the *shortest oscillation period* of the network so the delay risk is how much longer the longest orphan path delay is than the shortest orphan branch delay,
3. that the *orphan delay risk*, the shortest orphan branch delay subtracted from the longest orphan branch delay, is considerably less than the shortest period of the oscillation network encompassing the orphans ensuring that the longest orphan branch delay will complete its transition to **N** before the next wavefront of transition to **D** can arrive at the orphan branch.

Orphan branches are local and are a component of the oscillation period. A too long orphan branch delay can be contrived but, assuming a local uniformity of manufacturing variations over branches and behaviors, all local component delays will vary similarly and the orphan branch delays will always be proportionally shorter than the oscillation period. In Figure G.1 the longest green path has to propagate faster than the shortest purple path.

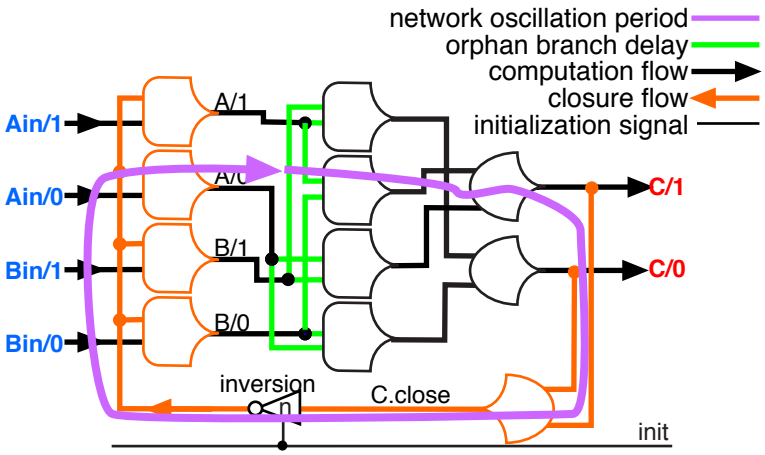


Figure G.1. Orphan risk: shortest oscillation period to longest orphan branch delay.

Orphan delays are timing relations relative to local delays within the network in contrast to timing relations critically relative to a global external time referent or to some indeterminate environmental delay (isochronic fork). It is generally sufficient to ensure that all orphan branches do not propagate through a primitive behavior to mitigate the orphan delay risk.

Appendix H: More Roman Numeral addition

H

The examples in this appendix making use of many intermediate differentness conditions, many more interactions and more complex dependency relations count the differentness conditions present in the Roman numeral addition entirely in term of interaction propensities among two or three differentness conditions instead of each differentness condition sensing a quorum of other differentness conditions present as with the interaction propensities of section 2.4.

There are two examples. Roman numeral addition A and Roman numeral addition B each with a different interaction protocol of counting the differentness condition present in the numerals of the addition. Both examples use buffer conditions for completeness of expression, propagate carries from digit group to digit group and determine when the addition is done.

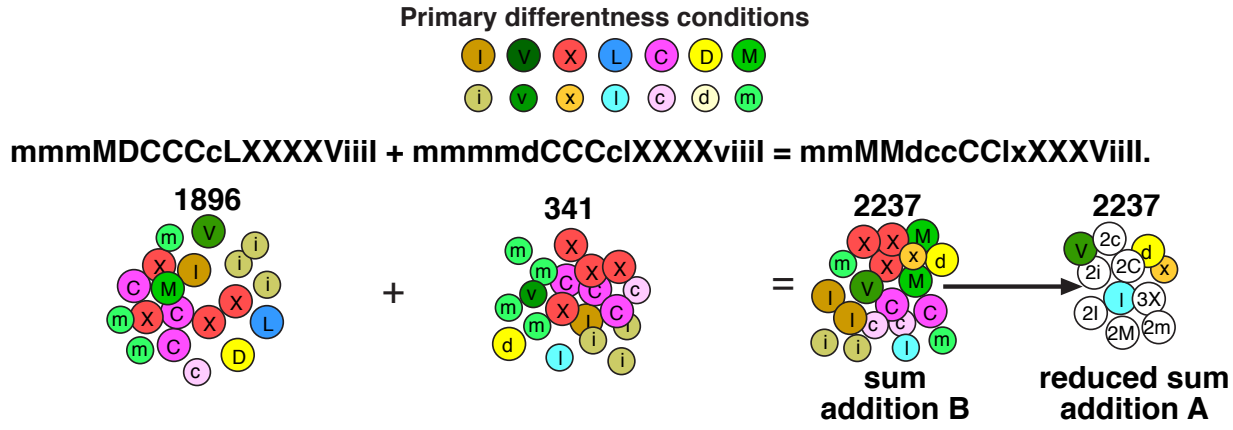


Figure H.1. The differentness conditions and the number representations.

H.1. Roman numeral addition A

The numeral differentness condition interact concurrently among themselves counting themselves with intermediate differentness conditions The addition is begun with two numbers present in the bag.

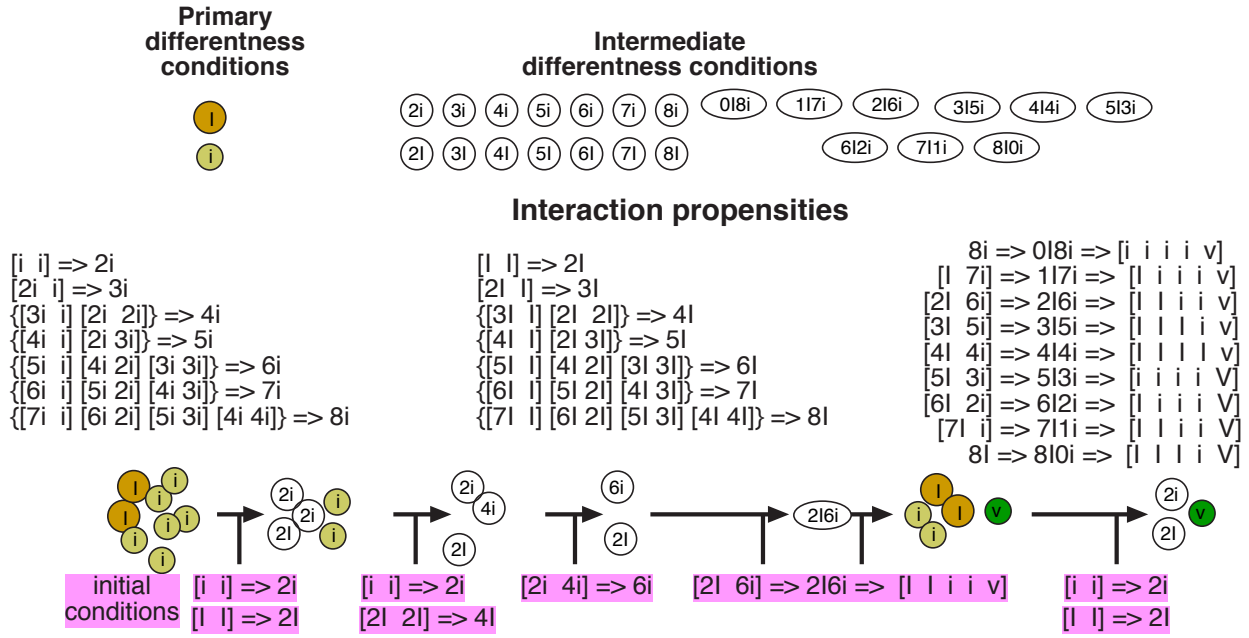


Figure H.2. The I/i interactions for Roman numeral addition A.

At the point where the sum is produced all of the initial differentness condition have disappeared so there is no ambiguity between the initial conditions and the result conditions. Because the conditions interact among themselves the sum differentness conditions will interact and reduce to a minimal form with intermediate differentness conditions. When a next numeral becomes present, even if it is also reduced to intermediate differentness conditions the two reduced numerals will interact to produce the correct new sum.

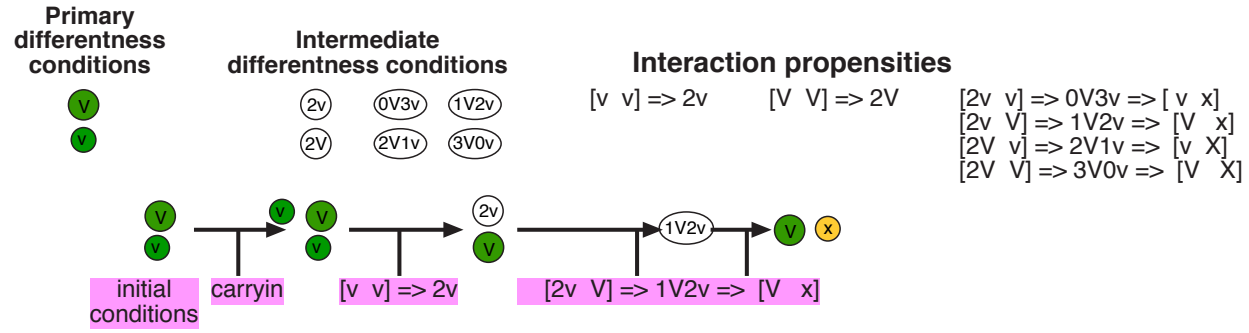


Figure H.3. The V/v interactions for Roman numeral addition A.

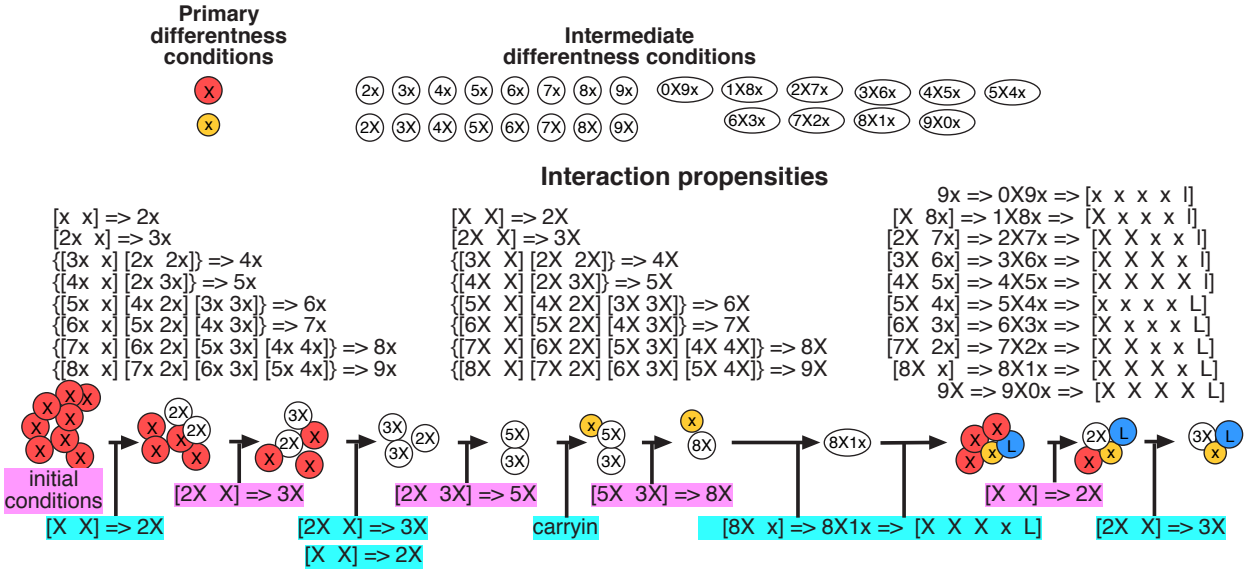


Figure H.4. The X/x interactions for Roman numeral addition A.

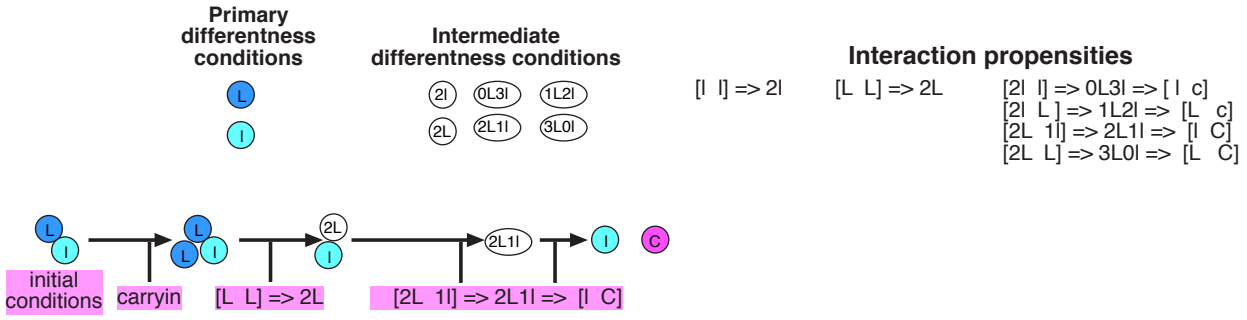


Figure H.5. The L/l interactions for Roman numeral addition A.

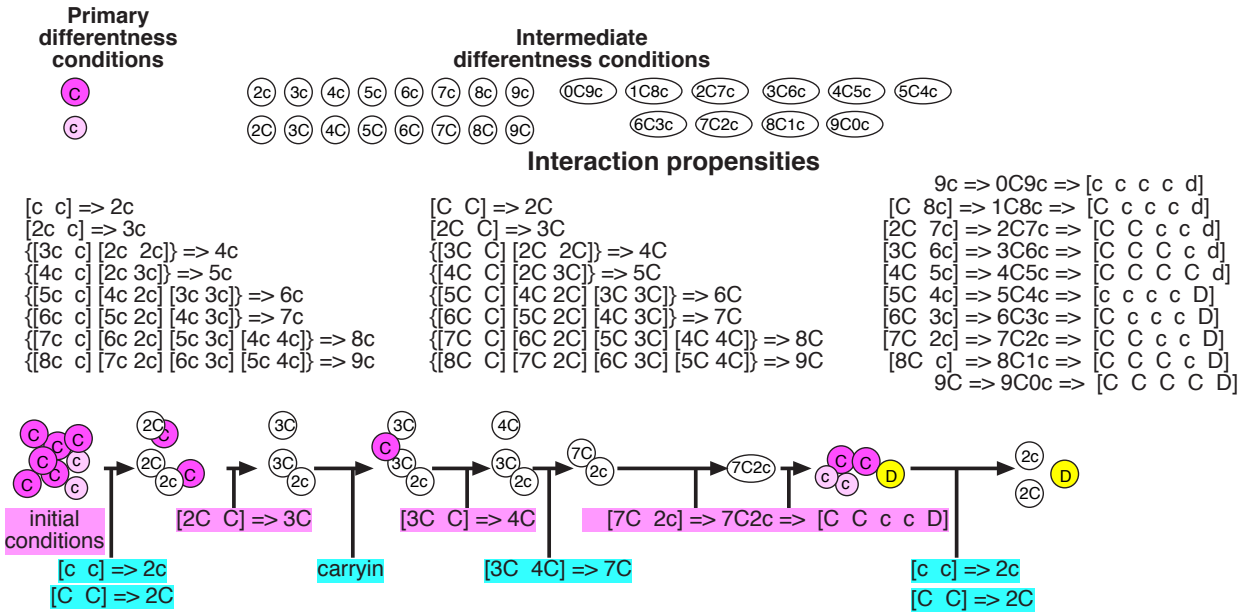


Figure H.6. The C/c interactions for Roman numeral addition A.

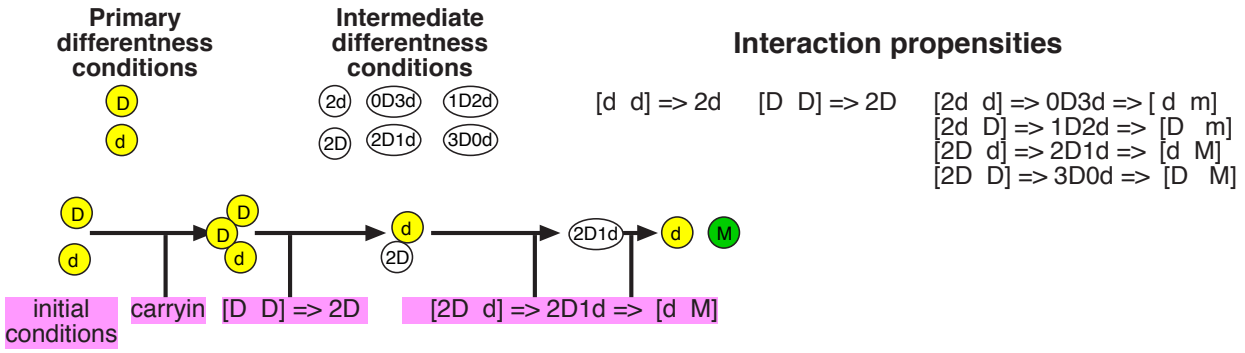


Figure H.7. The D/d interactions for Roman numeral addition A.

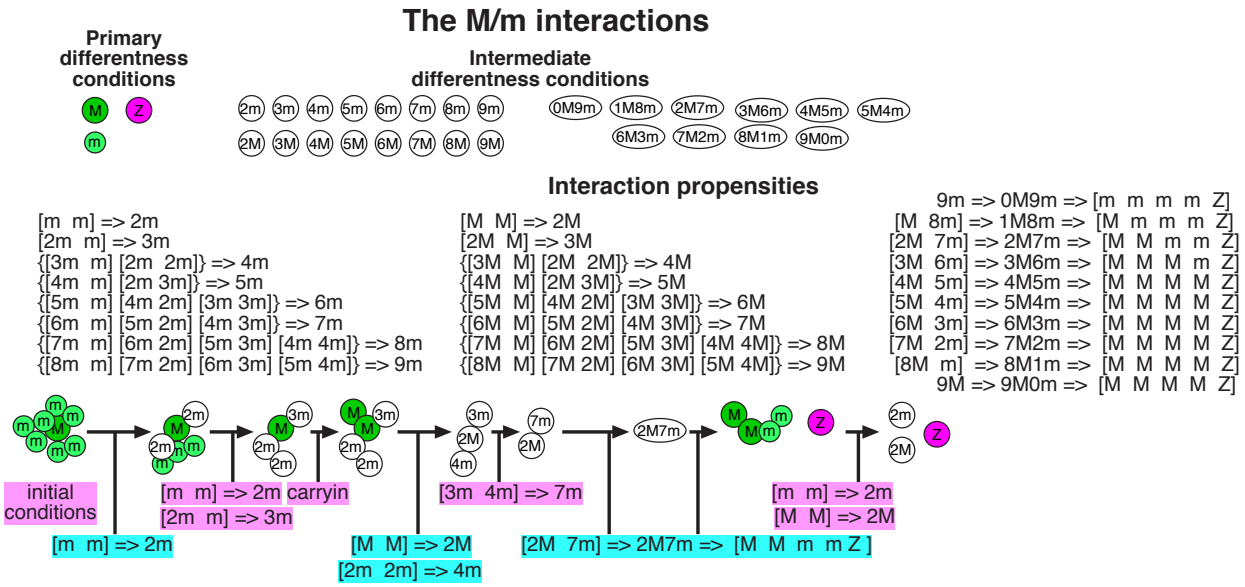


Figure H.8. The M/m interactions for Roman numeral addition A.

The **Z** differentness condition indicates the completion of the addition.

H.2. Roman numeral addition B

The numeral differentness conditions are counted sequentially with a starter control differentness condition. The addition is begun with two numbers present in the bag and a Starter differentness condition **S**.

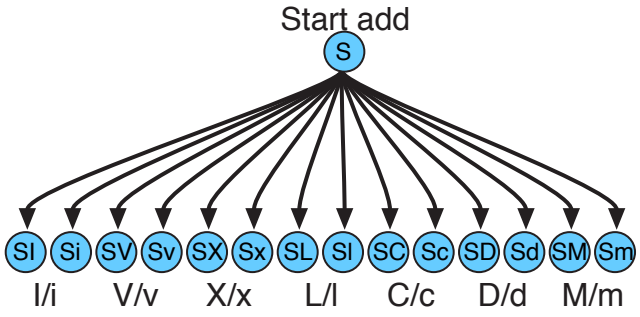


Figure H.9. The start differentness conditions are generated for all digit groups.

All seven digit groups begin counting their primary differentness conditions using intermediate differentness conditions. When the each count is determined the digit group result is produced with the carry to the next higher digit group. When the **M/m** digit group is completed the result number is completed and the **Z** condition is produced indicating the completion of the addition.

The complete addition is presented in Figure H.10 through Figure H.16 below.

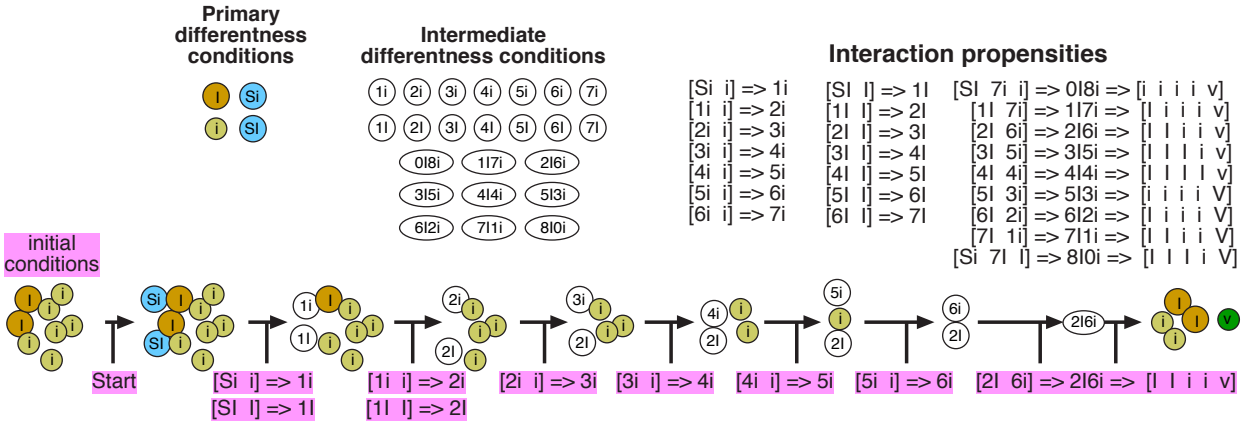


Figure H.10. The I/i interactions for Roman numeral addition B.

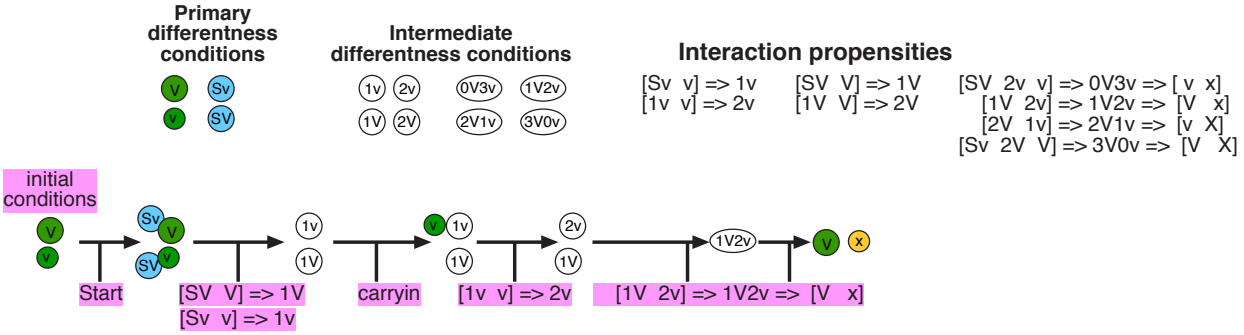


Figure H.11. The V/v interactions for Roman numeral addition B.

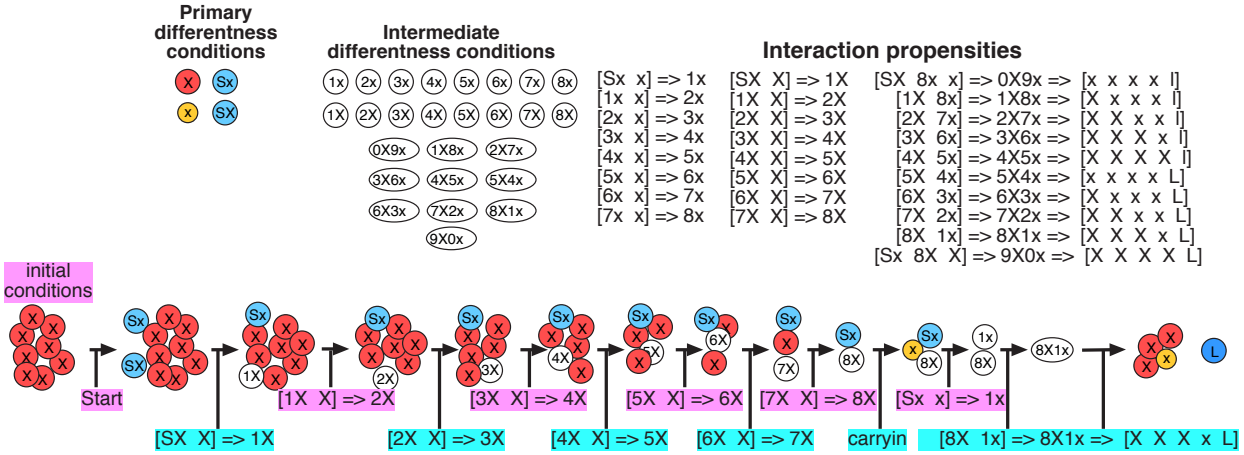


Figure H.12. The X/x interactions for Roman numeral addition B.

The Sx start differentness conditions did not interact with an x so must be included in the final interaction so that it disappears and does not ambiguate the next addition in the bag.

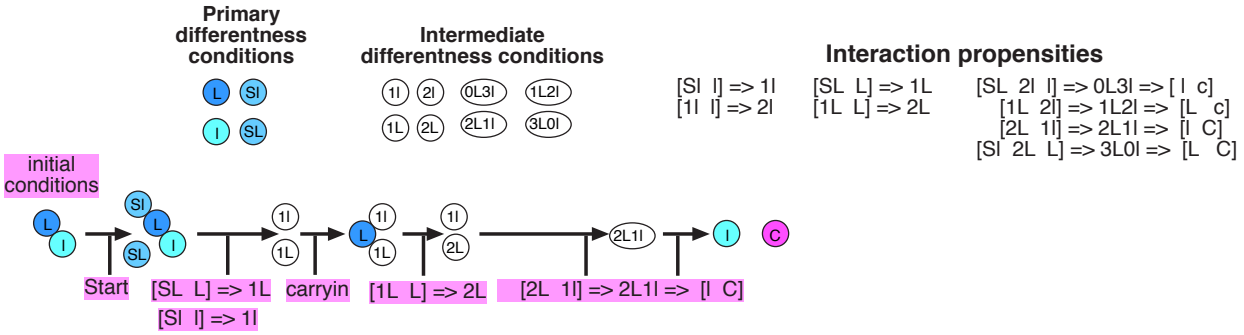


Figure H.13. The L/l interactions for Roman numeral addition B.

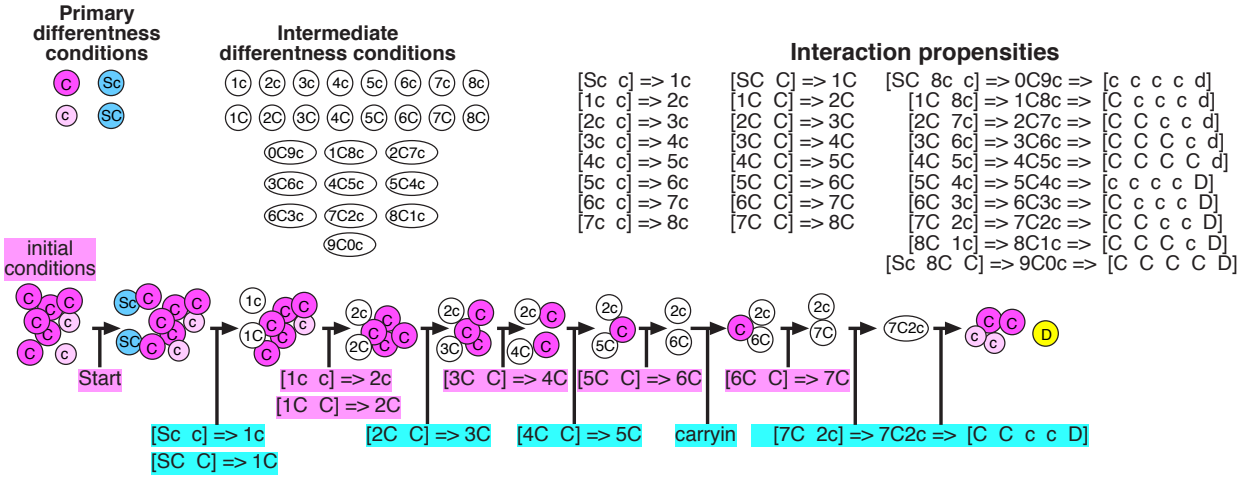


Figure H.14. The C/c interactions for Roman numeral addition B.

The C/c interactions.

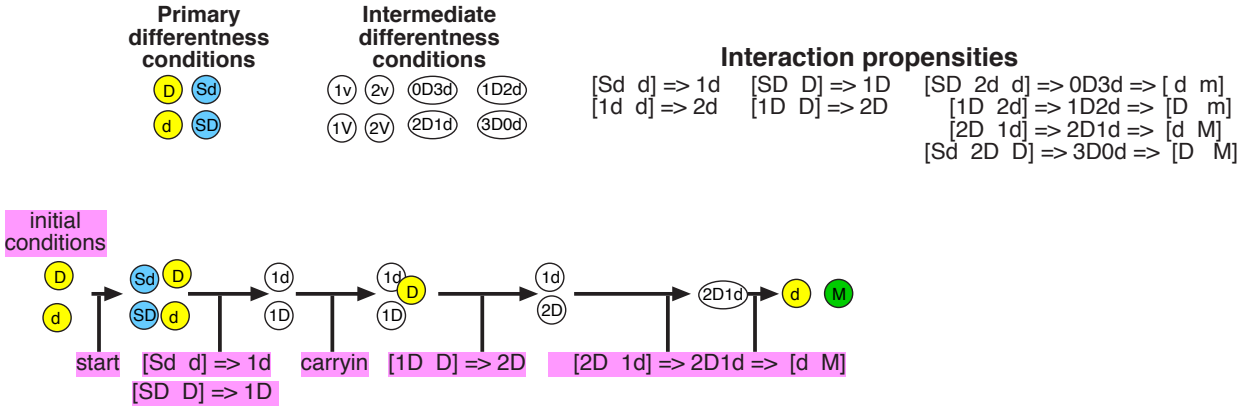


Figure H.15. The D/d interactions for Roman numeral addition B.

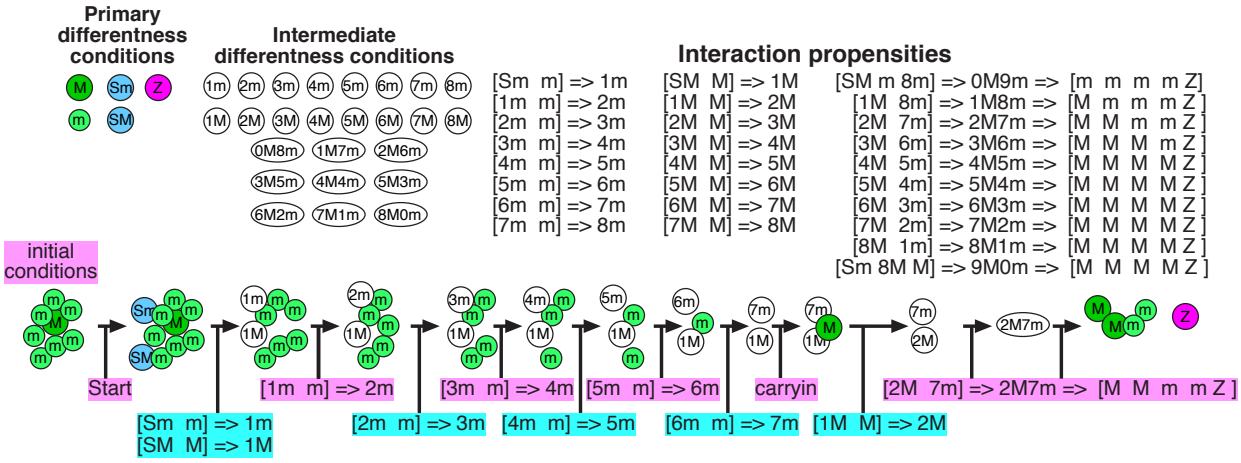
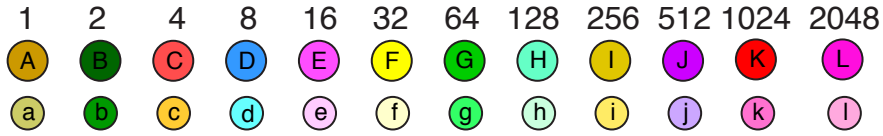


Figure H.16. The M/m interactions for Roman numeral addition B.

H.3. Roman numeral style binary addition

Binary numbers can be represented and added in Roman numeral form. Each differentness condition represents a different multiple of 2. Each multiple is represented by a unique differentness condition instead of being represented by differentness of association position.



11101101000 + 101010101 = 100010111101.
1896 + 341 = 2237

Figure H.17. Primary differentness conditions for binary number representation.

H.3.1. With buffer conditions

In relation to the convention binary number of 1 and 0 the higher case letters for each group are substituted for a 1 indicating that the multiple of 2 is present and the lower case letters are substituted for 0 indicating that the multiple of 2 is not present.

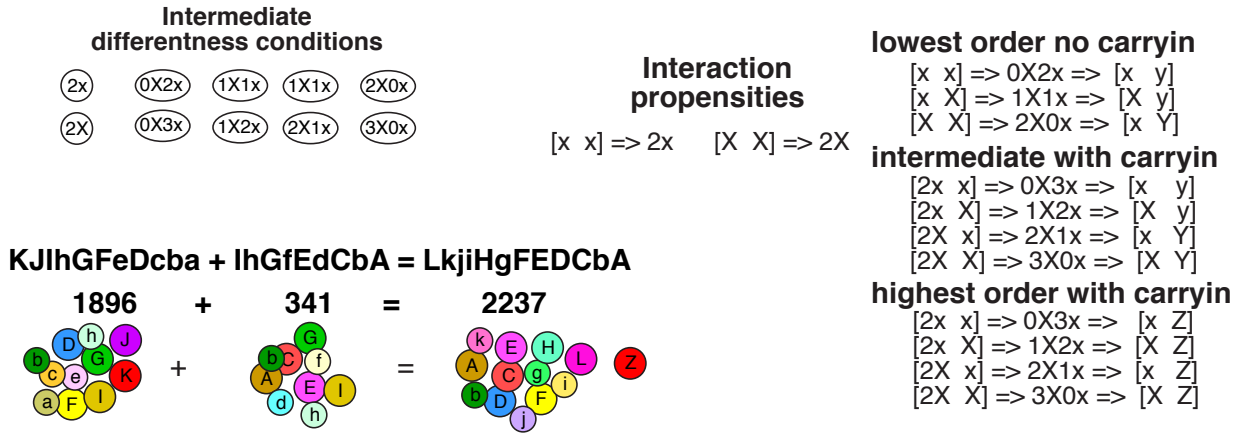


Figure H.18. Binary number with buffer differentness conditions.

The structure of the interaction propensities is the same for all digit groups. The differentness conditions and propensities for each digit group can be generated by substituting the **X/x** of Figure H.18with the letters of the group and the **Y/y** with the letters of the next higher order group. The lowest order interaction propensities do not include carryin but generate carryout. The intermediate interaction propensities include both carryin and carryout. The highest order interaction propensities include carryin but generate the completeness condition **Z**.

H.3.2. Without buffer conditions

The capital letter represents the presence of the multiple of 2. The absence of a multiple of 2 is not represented it is just absent. Without buffer conditions, similarly to Figure 2.7, the addition interaction cannot indicate its own completion.

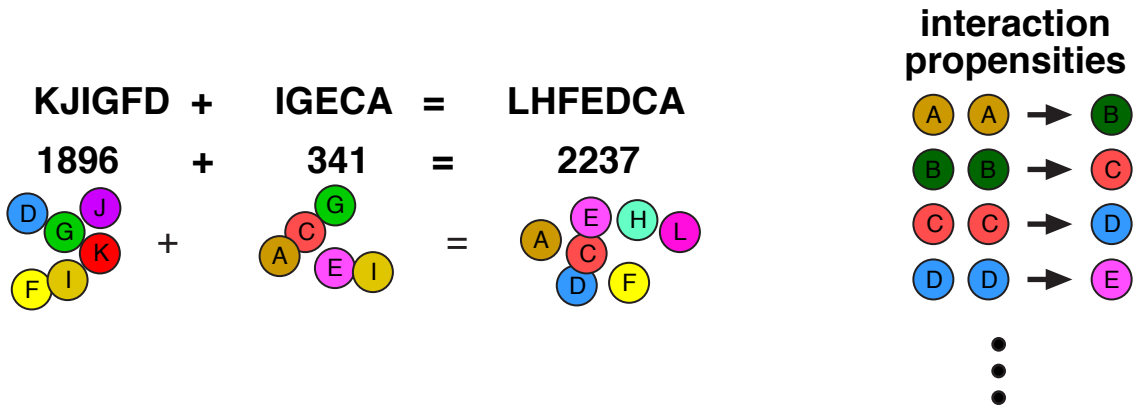


Figure H.19. Binary number without buffer differentness conditions.

Appendix I: Posts

I.1. State: A profoundly influential idea, but

While the notion of state has been a profound influence on intellectual progress as a basis of mathematical rigor and mechanism of truth it has also been a barrier to progress. The problem is that for any behavior to which state space is being applied the behavior must conform to the demands of the state space. Put another way, reality must conform to the demands of the model instead of the model conforming to the actualities of reality; concurrent behaviors, for instance, must be sequenced. This is why computer science is a diaphanous castle in the clouds with no founding in reality.

I.1.1. The notion of state and state space

A state is a particular condition of differentness from a range of possible conditions of differentness of a particular manifestation of state. There may be many particular manifestations of state forming a space of manifest states. Typically the range of possible conditions of differentness are shared among manifestations of state, numeric values for instance, allowing the possibility of identical conditions of differentness among states. Condition of differentness does not uniquely individuate a state so each manifestation of state is further individuated with a unique identity in the state space which may or may not involve a notion of structure of the state space. This state space individuation of each manifestation of state does not change while the content of each state can change. A state sample is of all manifestations of state in the state space with each's condition of differentness.

I.1.2. A state space must be constant

The state space remains constant with a population of state manifestations retaining their unique spatial individuation. It is the constancy of the state space that allows an external sampling agency to orient to the internal progress of the change behavior and to inspect the state of the state space state manifestation by state manifestation. There must be a referential constancy between the change behavior and the sampling agency.

The imposition

A change behavior can change the condition of differentness of state manifestations but may not change the spatial individuation of the state manifestations.

I.1.3. A state space must be determinate

To have a predicted state to compare to a sampled state the change behavior must be deterministic in relation to the state space.

The imposition

A change behavior cannot change states with indeterminate conditions of differentness.

I.1.4. A sampled state space must be stable

A state space can be reliably sampled only when all of its states are stable, i.e.,when no state is in the midst of change. A sampling agency must be able to discern from the change behavior when a stable sample can be taken or it must control the change behavior to establish a stable samplable state space. The change behavior may be halted and its internal state space inspected.

Or the internal state space may be extracted from the change behavior and inspected externally. In either case the sampling must be of a stable state space relative to the change behavior.

The imposition

The change behavior must explicitly make available intervals of stable state or it must submit to an external control that can establish a stable state space. The first order approach is for a change behavior to be one state change "at a time" sequential in that each change is completed before the next change is begun. Halted after the completion of any state change the change behavior presents a stable state until it is restarted. A change behavior can halt itself or the external agency can halt it.

I.1.5. State space sampling must be coordinated

If there is a predicted state to compare to a sampled state the sampling must be specifically coordinated to the progress of the change behavior. This requires coordination between the sampling agency and the change behavior.

The imposition

A change behavior must accommodate the coordination of sampling by providing a behavior discernible by the sampling agency at the appropriate state change.

I.1.6. There can be no concurrency

If a change behavior changes multiple states "at any time" it presents concurrent uncoordinated state change with no intrinsically discernible instants of stable state that offer opportunity for sampling and with no ordered occurrence of state changes that allows coordination of any specific state change to a specific sampled state.

The implication

The notion of state space fails in the context of concurrent change behavior. A concurrent change behavior clearly has internal manifestations of conditions of differentness and just as clearly the notion of state and state space cannot account them.

The imposition

Concurrent behavior cannot be allowed. All concurrent behavior must be mapped to sequential behavior.

I.1.7. The convenient street lamp

Computer science has never confronted concurrency directly but has always tried to characterize it in terms of sequentiality, that convenient street lamp at the end of the dark alley discouraging one from looking into the alley.

I.1.8. Internalization

Conforming all change behaviors to these impositions of observational needs leads to an internal samplable state space becoming viewed as endemic to all change behaviors whether the behavior might be sampled or whether it is even samplable. This internalization of state space is embodied by the Turing machine in its state machine and its paper tape. Change behaviors that do not conform, such as concurrent behaviors and neural networks do not qualify.

I.1.9. The humans in the works

