



Refinement Calculus in Type Theory

Peter Hancock
Nottingham
 Hancock@spamcop.net

EffTT, Tallinn, 13-14 Dec 2007

Refinement Calculus

Type Theory

Tentativity

A calculus of monotone predicate transformers

$$\Phi : \mathcal{P}(S') \rightarrow \mathcal{P}(S)$$

Semantics of a command:

If for any postcondition $U : \mathcal{P}(S')$ we know which preconditions $\Phi(U) : \mathcal{P}(S)$ will guarantee termination in a final state satisfying the postcondition, then we know the meaning of the command.

A calculus of monotone predicate transformers

$$\Phi : \mathcal{P}(S') \rightarrow \mathcal{P}(S)$$

Semantics of a command:

If for any postcondition $U : \mathcal{P}(S')$ we know which preconditions $\Phi(U) : \mathcal{P}(S)$ will guarantee termination in a final state satisfying the postcondition, then we know the meaning of the command.

A wide-spectrum ($Spec \sqsubseteq \dots \sqsubseteq Imp$) algebra of 2-party contracts, between an angel and a demon.



About 30-40 years old

- ▶ Floyd '67, Hoare '69. Flowchart/code annotations. (Partial correctness.)

About 30-40 years old

- ▶ Floyd '67, Hoare '69. Flowchart/code annotations. (Partial correctness.)
- ▶ Dijkstra '75. Guarded commands, weakest (liberal) precondition. [De Bakker, & De Roever '72, & Meertens '75]



About 30-40 years old

- ▶ Floyd '67, Hoare '69. Flowchart/code annotations. (Partial correctness.)
- ▶ Dijkstra '75. Guarded commands, weakest (liberal) precondition. [De Bakker, & De Roever '72, & Meertens '75]
- ▶ R-J Back '78 (Thesis, etc). Total correctness. Hehner.



About 30-40 years old

- ▶ Floyd '67, Hoare '69. Flowchart/code annotations. (Partial correctness.)
- ▶ Dijkstra '75. Guarded commands, weakest (liberal) precondition. [De Bakker, & De Roever '72, & Meertens '75]
- ▶ R-J Back '78 (Thesis, etc). Total correctness. Hehner.
- ▶ '87. Morgan, Morris, Nelson, Hesselink, R-J Back and von Wright.

About 30-40 years old

- ▶ Floyd '67, Hoare '69. Flowchart/code annotations. (Partial correctness.)
- ▶ Dijkstra '75. Guarded commands, weakest (liberal) precondition. [De Bakker, & De Roever '72, & Meertens '75]
- ▶ R-J Back '78 (Thesis, etc). Total correctness. Hehner.
- ▶ '87. Morgan, Morris, Nelson, Hesselink, R-J Back and von Wright.
- ▶ '98. Back and von Wright's Big Book.

About 30-40 years old

- ▶ Floyd '67, Hoare '69. Flowchart/code annotations. (Partial correctness.)
- ▶ Dijkstra '75. Guarded commands, weakest (liberal) precondition. [De Bakker, & De Roever '72, & Meertens '75]
- ▶ R-J Back '78 (Thesis, etc). Total correctness. Hehner.
- ▶ '87. Morgan, Morris, Nelson, Hesselink, R-J Back and von Wright.
- ▶ '98. Back and von Wright's Big Book.
- ▶ ...: parallel, reactive, object orientation ... data refinement/simulation ...

A trinity of categories

Gardiner, Martin, de Moore (1994)

- ▶ Sets and functions $f : A \rightarrow B$.

A trinity of categories

Gardiner, Martin, de Moore (1994)

- ▶ Sets and functions $f : A \rightarrow B$.
- ▶ Sets and relations $f : A \rightarrow \mathcal{P}(B)$. (Order enriched.)

A trinity of categories

Gardiner, Martin, de Moore (1994)

- ▶ Sets and functions $f : A \rightarrow B$.
- ▶ Sets and relations $f : A \rightarrow \mathcal{P}(B)$. (Order enriched.)
- ▶ Sets and (monotone) predicate transformers $f : A \rightarrow \mathcal{P}^2(B)$.

$$\begin{aligned} & A \rightarrow \mathcal{P}^2(B) \\ \simeq & \mathcal{P}(B) \rightarrow \mathcal{P}(A) \\ \simeq & \mathcal{P}(A \times \mathcal{P}(B)) \end{aligned}$$

Refinement order = pointwise inclusion.

A trinity of categories

Gardiner, Martin, de Moore (1994)

- ▶ Sets and functions $f : A \rightarrow B$.
- ▶ Sets and relations $f : A \rightarrow \mathcal{P}(B)$. (Order enriched.)
- ▶ Sets and (monotone) predicate transformers $f : A \rightarrow \mathcal{P}^2(B)$.

$$\begin{aligned} & A \rightarrow \mathcal{P}^2(B) \\ \simeq & \mathcal{P}(B) \rightarrow \mathcal{P}(A) \\ \simeq & \mathcal{P}(A \times \mathcal{P}(B)) \end{aligned}$$

Refinement order = pointwise inclusion.

- ▶ \downarrow . Skew-span. ‘Weak pullovers’. (Lax weak pullbacks.)

A trinity of categories

Gardiner, Martin, de Moore (1994)

- ▶ Sets and functions $f : A \rightarrow B$.
- ▶ Sets and relations $f : A \rightarrow \mathcal{P}(B)$. (Order enriched.)
- ▶ Sets and (monotone) predicate transformers $f : A \rightarrow \mathcal{P}^2(B)$.

$$\begin{aligned} A &\rightarrow \mathcal{P}^2(B) \\ &\simeq \mathcal{P}(B) \rightarrow \mathcal{P}(A) \\ &\simeq \mathcal{P}(A \times \mathcal{P}(B)) \end{aligned}$$

Refinement order = pointwise inclusion.

- ▶ \downarrow . Skew-span. ‘Weak pullovers’. (Lax weak pullbacks.)
- ▶ \uparrow . Maps.

An algebra of contracts

- ▶ Lattice structure: \sqsubseteq , \sqcup_i , \sqcap_i , **abort**, **magic**.



An algebra of contracts

- ▶ Lattice structure: \sqsubseteq , \sqcup_i , \sqcap_i , **abort**, **magic**.
- ▶ Modal structure: $\langle R \rangle$, $[R]$, where $R \subseteq S \times S'$

$$\langle R \rangle, [R] : \mathcal{P}(S') \rightarrow \mathcal{P}(S)$$

$$\langle R \rangle U \triangleq \{s \mid \exists s'. s R s' \wedge U(s')\}$$

$$[R] U \triangleq \{s \mid \forall s'. s R s' \rightarrow U(s')\}$$

Both: $\langle f \rangle = [f] = \text{assignment}$. $\langle U \rangle$ assertion, $[U]$ assumption.

An algebra of contracts

- ▶ Lattice structure: \sqsubseteq , \sqcup_i , \sqcap_i , **abort**, **magic**.
- ▶ Modal structure: $\langle R \rangle$, $[R]$, where $R \subseteq S \times S'$

$$\langle R \rangle, [R] : \mathcal{P}(S') \rightarrow \mathcal{P}(S)$$

$$\langle R \rangle U \triangleq \{s \mid \exists s'. s R s' \wedge U(s')\}$$

$$[R] U \triangleq \{s \mid \forall s'. s R s' \rightarrow U(s')\}$$

Both: $\langle f \rangle = [f] = \text{assignment}$. $\langle U \rangle$ assertion, $[U]$ assumption.

- ▶ Sequential structure: **skip**, \wp .

An algebra of contracts

- ▶ Lattice structure: $\sqsubseteq, \sqcup_i, \sqcap_i$, **abort**, **magic**.
- ▶ Modal structure: $\langle R \rangle, [R]$, where $R \subseteq S \times S'$

$$\langle R \rangle, [R] : \mathcal{P}(S') \rightarrow \mathcal{P}(S)$$

$$\langle R \rangle U \triangleq \{s \mid \exists s'. s R s' \wedge U(s')\}$$

$$[R] U \triangleq \{s \mid \forall s'. s R s' \rightarrow U(s')\}$$

Both: $\langle f \rangle = [f] = \text{assignment}$. $\langle U \rangle$ assertion, $[U]$ assumption.

- ▶ Sequential structure: **skip**, \wp .
- ▶ $\Phi^*(U) = (\mu V) U \cup \Phi(V)$, $\Phi^\infty(U) = (\nu V) U \cap \Phi(V)$,
- ▶ Monoidal closed structure: $\otimes, \multimap, !$. (If there's time.)



An algebra of contracts

- ▶ Lattice structure: \sqsubseteq , \sqcup_i , \sqcap_i , **abort**, **magic**.
- ▶ Modal structure: $\langle R \rangle$, $[R]$, where $R \subseteq S \times S'$

$$\langle R \rangle, [R] : \mathcal{P}(S') \rightarrow \mathcal{P}(S)$$

$$\langle R \rangle U \triangleq \{s \mid \exists s'. s R s' \wedge U(s')\}$$

$$[R] U \triangleq \{s \mid \forall s'. s R s' \rightarrow U(s')\}$$

Both: $\langle f \rangle = [f] = \text{assignment}$. $\langle U \rangle$ assertion, $[U]$ assumption.

- ▶ Sequential structure: **skip**, \wp .
- ▶ $\Phi^*(U) = (\mu V) U \cup \Phi(V)$, $\Phi^\infty(U) = (\nu V) U \cap \Phi(V)$,
- ▶ Monoidal closed structure: \otimes , \multimap , $!$. (If there's time.)
- ▶ Many 'healthiness' conditions: conjunctive, strict, continuous,
...

Two notions of subset (at least)

- ▶ Set-valued function on X . (Contravariant.)

$$\mathcal{P} : \text{Type} \rightarrow \text{Type}$$

$$\mathcal{P}(X) \triangleq X \rightarrow \text{Set}$$

Two notions of subset (at least)

- ▶ Set-valued function on X . (Contravariant.)

$$\mathcal{P} : \text{Type} \rightarrow \text{Type}$$

$$\mathcal{P}(X) \triangleq X \rightarrow \text{Set}$$

- ▶ set-indexed function into X . (Covariant.)

$$\mathcal{F} : \text{Type} \rightarrow \text{Type}$$

$$\mathcal{F}(X) \triangleq (\Sigma I : \text{Set}) I \rightarrow X$$

Two notions of subset (at least)

- ▶ Set-valued function on X . (Contravariant.)
 $\mathcal{P} : \text{Type} \rightarrow \text{Type}$
 $\mathcal{P}(X) \triangleq X \rightarrow \text{Set}$
- ▶ set-indexed function into X . (Covariant.)
 $\mathcal{F} : \text{Type} \rightarrow \text{Type}$
 $\mathcal{F}(X) \triangleq (\Sigma I : \text{Set}) I \rightarrow X$
- ▶ Predicative: if A is a set, neither $\mathcal{P}(A)$ nor $\mathcal{F}(A)$ is a set.
(Quite important: we are going to *iterate* \mathcal{F} .)

Two notions of subset (at least)

- ▶ Set-valued function on X . (Contravariant.)

$$\mathcal{P} : \text{Type} \rightarrow \text{Type}$$

$$\mathcal{P}(X) \triangleq X \rightarrow \text{Set}$$

- ▶ set-indexed function into X . (Covariant.)

$$\mathcal{F} : \text{Type} \rightarrow \text{Type}$$

$$\mathcal{F}(X) \triangleq (\Sigma I : \text{Set}) I \rightarrow X$$

- ▶ Predicative: if A is a set, neither $\mathcal{P}(A)$ nor $\mathcal{F}(A)$ is a set.
(Quite important: we are going to *iterate* \mathcal{F} .)

'Set' = *data type ; given exhaustively by constructors (the operands of which need not be data ...)*.

Two notions of subset (at least)

- ▶ Set-valued function on X . (Contravariant.)

$$\mathcal{P} : \text{Type} \rightarrow \text{Type}$$

$$\mathcal{P}(X) \triangleq X \rightarrow \text{Set}$$

- ▶ set-indexed function into X . (Covariant.)

$$\mathcal{F} : \text{Type} \rightarrow \text{Type}$$

$$\mathcal{F}(X) \triangleq (\Sigma I : \text{Set}) I \rightarrow X$$

- ▶ Predicative: if A is a set, neither $\mathcal{P}(A)$ nor $\mathcal{F}(A)$ is a set.
(Quite important: we are going to *iterate* \mathcal{F} .)

'Set' = *data* type ; given exhaustively by constructors (the operands of which need not be data ...).

Representations of predicate transformers

- ▶ $\Phi : S \rightarrow \mathcal{F}^2(S')$. (Petersson and Synek, 1989.)

$$\Phi = (\lambda s) \langle C(s), (\lambda c) \langle R(s, c), (\lambda r) n(s, c, r) \rangle \rangle$$

C	$: S \rightarrow \text{Set}$	Commands
R	$: (\prod s : S) C(s) \rightarrow \text{Set}$	Responses
n	$: (\prod s : S, c : C(s)) R(s, c) \rightarrow S'$	next state

Representations of predicate transformers

- ▶ $\Phi : S \rightarrow \mathcal{F}^2(S')$. (Petersson and Synek, 1989.)

$$\Phi = (\lambda s) \langle C(s), (\lambda c) \langle R(s, c), (\lambda r) n(s, c, r) \rangle \rangle$$

C	$: S \rightarrow \text{Set}$	Commands
R	$: (\prod s : S) C(s) \rightarrow \text{Set}$	Responses
n	$: (\prod s : S, c : C(s)) R(s, c) \rightarrow S'$	next state

- ▶ Extension: $\|\Phi\| : \mathcal{P}(S') \rightarrow \mathcal{P}(S)$

$$U : \mathcal{P}(S') \mapsto \{s : S \mid (\sum c : C(s)) (\prod r : R(s, c)) U(n(s, c, r))\}$$

Eg, **abort**: $C(s) = \emptyset$; **magic**: $R(s, c) = \emptyset$.

Free monad

$$\Phi : S \rightarrow \mathcal{F}^2(S)$$

Free monad

$$\Phi : S \rightarrow \mathcal{F}^2(S)$$

$$\|\Phi^*\|X = (\mu Y) X \cup \Phi(X)$$



Free monad

$$\Phi : S \rightarrow \mathcal{F}^2(S)$$

$$\|\Phi^*\|X = (\mu Y) X \cup \Phi(X)$$

► Programs $C^*(s)$

$$\begin{aligned} C^*(s) &= 1 + (\Sigma c : C(s)) (\Pi r : R(s, c)) C^*(n(s, c, r)) \\ &= 1 + \|\Phi\|(C^*, s) \end{aligned}$$



Free monad

$$\Phi : S \rightarrow \mathcal{F}^2(S)$$

$$\|\Phi^*\|X = (\mu Y) X \cup \Phi(X)$$

- ▶ Programs $C^*(s)$

$$\begin{aligned} C^*(s) &= 1 + (\sum c : C(s)) (\prod r : R(s, c)) C^*(n(s, c, r)) \\ &= 1 + \|\Phi\|(C^*, s) \end{aligned}$$

- ▶ Traces $R^*(s, p)$ where $s : S, p : C^*(s)$:

$$\begin{aligned} R^*(s, \mathbf{Exit}) &= 1 \\ R^*(s, \mathbf{Call}(c, f)) &= (\sum r : R(s, c)) R^*(n(s, c, r), f(r)) \end{aligned}$$

Free monad

$$\Phi : S \rightarrow \mathcal{F}^2(S)$$

$$\|\Phi^*\|X = (\mu Y) X \cup \Phi(X)$$

- ▶ Programs $C^*(s)$

$$\begin{aligned} C^*(s) &= 1 + (\sum c : C(s)) (\prod r : R(s, c)) C^*(n(s, c, r)) \\ &= 1 + \|\Phi\|(C^*, s) \end{aligned}$$

- ▶ Traces $R^*(s, p)$ where $s : S, p : C^*(s)$:

$$\begin{aligned} R^*(s, \mathbf{Exit}) &= 1 \\ R^*(s, \mathbf{Call}(c, f)) &= (\sum r : R(s, c)) R^*(n(s, c, r), f(r)) \end{aligned}$$

- ▶ Exit state $n^*(s, p, t)$ where $s : S, p : C^*(s), t : R^*(s, p)$:

$$\begin{aligned} n^*(s, \mathbf{Exit}) &= 1 \\ n^*(s, \mathbf{Call}(c, f), \langle r, t \rangle) &= n^*(n(s, c, r), f(r), t) \end{aligned}$$

Morphisms

- ▶ Objects: *interaction structures*

$$\langle S : \text{Set}, \Phi : S \rightarrow \mathcal{F}^2(S) \rangle$$



Morphisms

- ▶ Objects: *interaction structures*

$$\langle S : \text{Set}, \Phi : S \rightarrow \mathcal{F}^2(S) \rangle$$

- ▶ Morphisms: *simulations*

$$\langle S : \text{Set}, \Phi : S \rightarrow \mathcal{F}^2(S) \rangle$$

$$\downarrow$$

$$\langle S' : \text{Set}, \Psi : S' \rightarrow \mathcal{F}^2(S') \rangle$$

given by $R \subseteq S' \times S$ such that

$$\langle R \rangle \cdot \|\Phi\| \sqsubseteq \|\Psi\| \cdot \langle R \rangle$$

What this means

$$\begin{aligned} R(s, s') \rightarrow & \Sigma \quad f : C(s) \rightarrow C'(s') \\ & g : (\Pi c : C(s)) R'(s', f(c)) \rightarrow R(s, c) \\ & \Pi \quad c : C(s) \\ & \quad r' : R'(s', f(c)) \\ & R(n(s, c, g(c, r')), n'(s', f(c), r')) \end{aligned}$$



What this means

$$\begin{aligned}
 R(s, s') \rightarrow & \Sigma \quad f : C(s) \rightarrow C'(s') \\
 & g : (\prod c : C(s)) R'(s', f(c)) \rightarrow R(s, c) \\
 & \Pi \quad c : C(s) \\
 & \quad r' : R'(s', f(c)) \\
 & R(n(s, c, g(c, r')), n'(s', f(c), r'))
 \end{aligned}$$

Er, hang on, that's a coalgebra for ... an interaction structure!
 (That's where some linear logic comes in...).



What this means

$$\begin{aligned}
 R(s, s') \rightarrow & \Sigma \quad f : C(s) \rightarrow C'(s') \\
 & \quad g : (\prod c : C(s)) R'(s', f(c)) \rightarrow R(s, c) \\
 & \Pi \quad c : C(s) \\
 & \quad r' : R'(s', f(c)) \\
 & \quad R(n(s, c, g(c, r')), n'(s', f(c), r'))
 \end{aligned}$$

Er, hang on, that's a coalgebra for ... an interaction structure!
 (That's where some linear logic comes in...).

Composition is relational composition. (This isn't great...)

What is the logical form of a specification?

What a silly question! It depends. . .

What is the logical form of a specification?

What a silly question! It depends. . .

- ▶ A *client* (transaction) program:

$$\underbrace{U}_{\text{Precondition}} \subseteq \Phi^* \left(\underbrace{V}_{\text{Postcondition}} \right)$$

What is the logical form of a specification?

What a silly question! It depends. . .

- ▶ A *client* (transaction) program:

$$\underbrace{U}_{\text{Precondition}} \subseteq \Phi^* \left(\underbrace{V}_{\text{Postcondition}} \right)$$

- ▶ A *server* program:

$$\underbrace{U}_{\text{Initialisation}} \wp (\Phi^\sim)^\infty \left(\underbrace{V}_{\text{Safety property}} \right)$$

$$\begin{aligned} \Phi^\sim \text{ is } \textit{inversion} \text{ of } \Phi: \quad C^\sim(s) &= (\prod c : C(s)) R(s, c) \\ R^\sim(s, -) &= C(s) \\ n^\sim(s, f, c) &= n(s, c, f(c)) \end{aligned}$$

Thinking more practically

- ▶ An *Instruction set*: $\langle I, O \rangle : \mathcal{F}(\text{Set})$

Thinking more practically

- ▶ An *Instruction set*: $\langle I, O \rangle : \mathcal{F}(\text{Set})$
- ▶ Specifications (man pages) in the 'real world' seem to have the form:

$$S \rightarrow (\prod i : I) \mathcal{F}(\mathcal{P}(O(i) \times S))$$

The data here is

$$P : \mathcal{P}(S \times I)$$

$$N : (\prod \langle s, i \rangle : S \times I) P \langle s, i \rangle \rightarrow \mathcal{P}(O(i) \times S)$$

A species of non-deterministic Mealey machine?

A relative of Lynch's IO-automata?

Some opinions (on 'the IO problem')

Some opinions (on 'the IO problem')

- ▶ We need new ideas. Probably several. Do not overlook any source. Linearity. Temporal logic. Topology. Alcohol.

Some opinions (on 'the IO problem')

- ▶ We need new ideas. Probably several. Do not overlook any source. Linearity. Temporal logic. Topology. Alcohol.
- ▶ We need old ideas. For example predicate transformers. Simulations. Streams. This ought to be reassuring.

Some opinions (on 'the IO problem')

- ▶ We need new ideas. Probably several. Do not overlook any source. Linearity. Temporal logic. Topology. Alcohol.
- ▶ We need old ideas. For example predicate transformers. Simulations. Streams. This ought to be reassuring.
- ▶ Stop obsessing about monads!

Some opinions (on 'the IO problem')

- ▶ We need new ideas. Probably several. Do not overlook any source. Linearity. Temporal logic. Topology. Alcohol.
- ▶ We need old ideas. For example predicate transformers. Simulations. Streams. This ought to be reassuring.
- ▶ Stop obsessing about monads!
- ▶ Stop obsessing about programs (terms). Start obsessing about *specifications*!

Some opinions (on 'the IO problem')

- ▶ We need new ideas. Probably several. Do not overlook any source. Linearity. Temporal logic. Topology. Alcohol.
- ▶ We need old ideas. For example predicate transformers. Simulations. Streams. This ought to be reassuring.
- ▶ Stop obsessing about monads!
- ▶ Stop obsessing about programs (terms). Start obsessing about *specifications*!
- ▶ Forget about computers. A program is a guide to action. Action directed towards an end.

A few references I



Ralph-Johan Back and Joakim von Wright.

Refinement calculus, A systematic introduction.

Graduate Texts in Computer Science. Springer-Verlag, New York, 1998.



Ralph-Johan Back and Joakim Von Wright.

Encoding, decoding and data refinement.

Formal Aspects of Computing, 12(5):313–349, 2000.



Paul H. B. Gardiner, Clare E. Martin, and Oege de Moor.

An algebraic construction of predicate transformers.

Science of Computer Programming, 22:21–44, 1994.



Peter Hancock and Pierre Hyvernat.

Programming interfaces and basic topology.

Ann. Pure Appl. Logic, 137(1-3):189–239, 2006.



Peter Hancock and Anton Setzer.

Interactive programs in dependent type theory.

In *Proceedings of the 14th Annual Conference of the EACSL on Computer Science Logic*, pages 317–331, London, UK, 2000. Springer-Verlag.



Peter Hancock and Anton Setzer.

Guarded induction and weakly final coalgebras in dependent type theory.

In L. Crosilla and P. Schuster, editors, *From Sets and Types to Topology and Analysis. Towards Practicable Foundations for Constructive Mathematics*, pages 115 – 134, Oxford, 2005. Clarendon Press.



A few references II



Pierre Hyvernat.

Predicate transformers and linear logic: yet another denotational model.

In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *18th International Workshop CSL 2004*, volume 3210 of *Lecture Notes in Computer Science*, pages 115–129. Springer-Verlag, September 2004.



Markus Michelbrink.

Interfaces as functors, programs as coalgebras: a final coalgebra theorem in intensional type theory.

Theor. Comput. Sci., 360(1):415–439, 2006.



Carroll Morgan.

Programming from Specifications.

Prentice Hall International Series in Computer Science, 2nd edition, 1994.



Markus Michelbrink and Anton Setzer.

State dependent IO-monads in type theory.

Electronic Notes in Theoretical Computer Science, 122:127 – 146, 2005.



Kent Petersson and Dan Synek.

A set constructor for inductive sets in Martin-Löf's type theory.

In *Proceedings of the 1989 Conference on Category Theory and Computer Science, Manchester, UK*, volume 389 of *LNCS*. Springer Verlag, 1989.



Papertape IO

$$\begin{aligned}
 S &= \text{Char}^\omega \times \text{Char}^* \\
 C \langle i, o \rangle &= \{\mathbf{Get}\} | \{\mathbf{Put } c \mid c : \text{Char}\} \\
 R \langle i, o \rangle \mathbf{Get} &= (\Sigma c : \text{Char}) c = \text{hd } i \\
 n \langle i, o \rangle \mathbf{Get} \langle c, _ \rangle &= \langle \text{tl } i, o \rangle \\
 R \langle i, o \rangle (\mathbf{Put } c) &= \{\mathbf{Ack}\} \\
 n \langle i, o \rangle (\mathbf{Put } c) \mathbf{Ack} &= \langle i, o ++ \langle c \rangle \rangle
 \end{aligned}$$



Nim

$$\begin{aligned}
 S &= Z \\
 C\ l &= (l = 0) + (l > 0) + (l > 1) \\
 R\ l\ (\mathbf{in}_0-) &= \{\} \\
 R\ l\ (\mathbf{in}_1-) &= C(l - 1) \\
 R\ l\ (\mathbf{in}_2-) &= C(l - 2) \\
 n\ l\ (\mathbf{in}_1-)\ (\mathbf{in}_0-) &= -1 \\
 n\ l\ (\mathbf{in}_1-)\ (\mathbf{in}_1-) &= l - 2 \\
 n\ l\ (\mathbf{in}_1-)\ (\mathbf{in}_2-) &= l - 3 \\
 n\ l\ (\mathbf{in}_2-)\ (\mathbf{in}_0-) &= -1 \\
 n\ l\ (\mathbf{in}_2-)\ (\mathbf{in}_1-) &= l - 3 \\
 n\ l\ (\mathbf{in}_2-)\ (\mathbf{in}_2-) &= l - 4
 \end{aligned}$$



Nim

$$\begin{aligned}
 S &= Z \\
 C\ l &= (l = 0) + (l > 0) + (l > 1) \\
 R\ l\ (\mathbf{in}_0-) &= \{\} \\
 R\ l\ (\mathbf{in}_1-) &= C(l - 1) \\
 R\ l\ (\mathbf{in}_2-) &= C(l - 2) \\
 n\ l\ (\mathbf{in}_1-)\ (\mathbf{in}_0-) &= -1 \\
 n\ l\ (\mathbf{in}_1-)\ (\mathbf{in}_1-) &= l - 2 \\
 n\ l\ (\mathbf{in}_1-)\ (\mathbf{in}_2-) &= l - 3 \\
 n\ l\ (\mathbf{in}_2-)\ (\mathbf{in}_0-) &= -1 \\
 n\ l\ (\mathbf{in}_2-)\ (\mathbf{in}_1-) &= l - 3 \\
 n\ l\ (\mathbf{in}_2-)\ (\mathbf{in}_2-) &= l - 4
 \end{aligned}$$

$$\begin{aligned}
 &(\mu X)\ [l > 0] \ ; \langle M \rangle \ ; \langle l > 0 \rangle \ ; [M] \ ; X \\
 &\mathbf{where}\ M \triangleq (l := l' \mid l - 2 \leq l' \leq l - 1)
 \end{aligned}$$



Some programming constructs

$$\begin{aligned}
 \|\mathbf{skip}\| U &= U \\
 \|P \mathbin{;} Q\| &= \|P\| \cdot \|Q\| \\
 \|\mathbf{abort}\| U &= \mathit{false} \\
 \|\mathbf{magic}\| U &= \mathit{true} \\
 \|v := e\| U &= U[v \leftarrow e] \\
 \|\langle P \rangle\| U &= P \wedge U \\
 \|[P]\| U &= P \rightarrow U \\
 \|\langle R \rangle\| U &= \{s \mid R(s) \not\subseteq U\} \\
 \|[R]\| U &= \{s \mid R(s) \subseteq U\} \\
 \|\sqcup_i P_i\| U &= \bigvee_i \|P_i\| U \\
 \|\sqcap_i P_i\| U &= \bigwedge_i \|P_i\| U
 \end{aligned}$$