Contents

Legacy

Swinging Types inherit from

algebraic specification

many-sorted logic

functional and logic programming

automated theorem proving

term rewriting

structural operational semantics

relational fixpoint theory $A \subseteq B$

category theory $A \to B$

universal algebra $F(A) \to A$

universal coalgebra $A \to F(A)$

modal logic

Objektorientierte Spezifikation und Verifikation mit Coalgebren und Coinduktion

Peter Padawitz
University of Dortmund

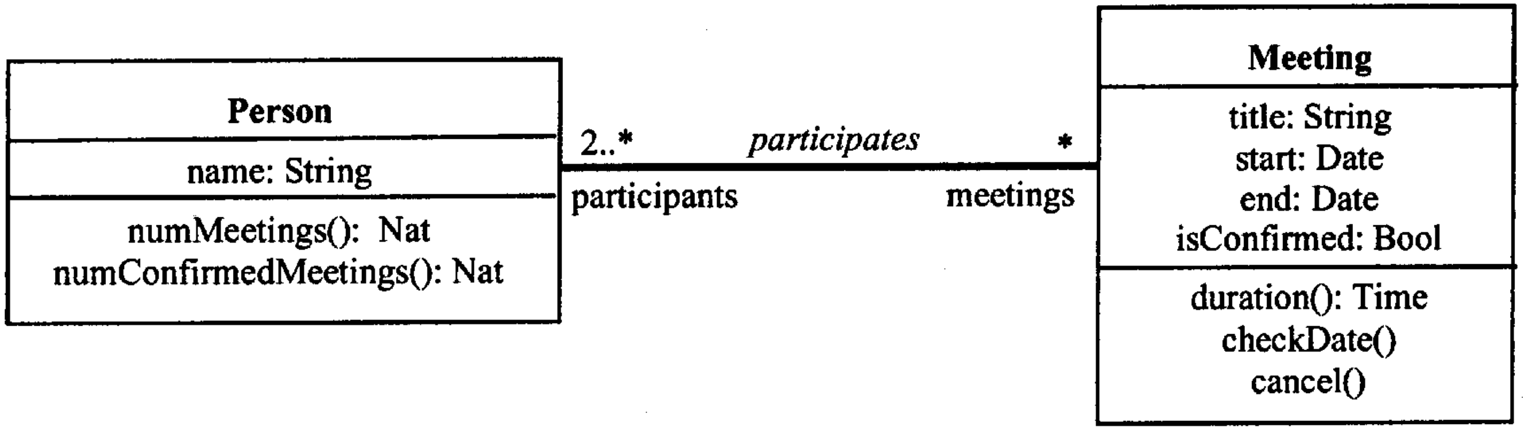
ls5.cs.uni-dortmund.de/~peter/Swinging.html
ls5.cs.uni-dortmund.de/~peter/Expander2.html

Rurberg 2001

1. Swinging types, informally

2. Specifications and Cospecifications

3. Induction and Coinduction



Two associated classes

from: H. Hußmann, M. Cerioli, G. Reggio, F. Tort, Abstract Data Types and UML Models, Report DISI-TR-99-15, University of Genova 1999

OCL constraint:

context Meeting :: checkDate()
post : isConfirmed =
self.participants ->
collect(meetings) ->
forAll(m | m <> self and
m.isConfirmed implies
(after(self.end,m.start) or
(after(m.end,self.start)))

PERSON&MEETING $=$ ... `then`

`hidsorts`	$Person \quad Meeting$
`destructs`	$name : Person \to String$
	$meetings : Person \to Meeting^*$
	$title : Meeting \to String$
	$participants : Meeting \to Person^*$
	$start, end : Meeting \to Date$
	$isConfirmed : Meeting \to bool$
`constructs`	$checkDate : Meeting \to Meeting$
	$cancel : Meeting \to Meeting$
`defuncts`	$numMeetings : Person \to nat$
	$Meetings : Person \to set(Meeting)$
	$Participants : Meeting \to set(Person)$
	$numConfirmedMeetings : Person \to nat$
	$duration : Meeting \to Time$
	$consistent : Meeting \times Meeting \to bool$
`vars`	$p : Person \quad m, m' : Meeting \quad ms : set(Meeting)$
	$ps : set(Person)$
`Horn axioms`	$Meetings(p) \ \equiv \ mkset(meetings(p))$
	$Participants(p) \ \equiv \ mkset(participants(m))$
	$numMeetings(p) \ \equiv \ \lvert Meetings(p) \rvert$
	$numConfirmedMeetings(p)$
	$\quad \equiv \ \lvert filter(isConfirmed, Meetings(p)) \rvert$
	$duration(m) \ \equiv \ end(m) - start(m)$
	$consistent(m, m')$
	$\quad \equiv \ not(isConfirmed(m'))$
	$\qquad or \ end(m) < start(m') \ or \ end(m') < start(m)$
	$isConfirmed(checkDate(m))$
	$\quad \equiv \ forall(\lambda m'.consistent(m, m'), remove(m, ms))$
	$\qquad \Leftarrow \ Participants(m) \equiv ps \ \wedge$
	$\qquad\quad unionMap(Meetings)(ps) \equiv ms$
	$isConfirmed(cancel(m)) \ \equiv \ false$
`assertions`	$Participants(m) \rvert \geq 2$

Each **generalization** arrow pointing from a subclass cl' to a direct superclass cl of cl' yields a constructor

$$in^{cl}_{cl'} : cl' \to cl$$

of $SP(cl)$.

Let $<$ be the inheritance relation associated with the class diagram and the nodes form a finite lattice w.r.t. $<$: several classes that inherit to the same subclass have a common superclass.

The specifications of classes related by $<$ must be augmented with **frame axioms** for methods/attributes that are defined functions. Let cl' be a direct subclass of cl.

- **Adding subclass properties of cl'.** Let $f_{cl} : cl \times s_1 \times \cdots \times s_n \to cl$ and $g_{cl} : cl \times s_1 \times \cdots \times s_n \to s$, $s \neq cl$, be used, but not redefined in $SP(cl')$. Then the following axioms are added to $SP(cl')$:

$$\begin{array}{lcl} f_{cl'}(x, x_1, \ldots, x_n) \equiv y & \Leftarrow & f_{cl}(in^{cl}_{cl'}(x), x_1, \ldots, x_n) \equiv in^{cl}_{cl'}(y) \\ g_{cl'}(x, x_1, \ldots, x_n) \equiv y & \Leftarrow & g_{cl}(in^{cl}_{cl'}(x), x_1, \ldots, x_n) \equiv y. \end{array}$$

- **Adding superclass properties of cl.** Let $f_{cl'} : cl' \times s_1 \times \cdots \times s_n \to cl'$ and $g_{cl'} : cl' \times s_1 \times \cdots \times s_n \to s$, $s \neq cl$, be defined in $SP(cl')$, but not in $SP(cl)$. Then the following axioms are added to $SP(cl)$:

$$\begin{array}{l} f_{cl}(in^{cl}_{cl'}(x), x_1, \ldots, x_n) \equiv in^{cl}_{cl'}(f_{cl'}(x, x_1, \ldots, x_n)) \\ g_{cl}(in^{cl}_{cl'}(x), x_1, \ldots, x_n) \equiv g_{cl'}(x, x_1, \ldots, x_n). \end{array}$$

Multiple inheritance. Let cl_0 be a common superclass of cl_1 and cl_2 and cl_1 and cl_2 be superclasses of cl_3. Then a cl_3-object x may have two normal form representations in cl_0:

$$t \;=\; in_{cl_1}^{cl_0}(in_{cl_3}^{cl_1}(x)) \quad \text{and} \quad t' \;=\; in_{cl_2}^{cl_0}(in_{cl_3}^{cl_2}(x)).$$

For keeping track of the unique origin of an object, each class cl is extended by an equivalence relation $\approx_{cl}: cl \times cl$ and a predicate $\approx_{cl'}^{cl}: cl' \times cl$ satisfying the axioms

$$\begin{array}{rcl} in_{cl'}^{cl}(x) \approx_{cl} y & \Leftarrow & x \approx_{cl'}^{cl} y \\ x \approx_{cl'}^{cl} in_{cl'}^{cl}(y) & \Leftarrow & x \approx_{cl'} y \end{array}$$

for each direct subclass cl' of cl.

In the entire class diagram, $\approx_{cl}$ provides an equality for cl that identifies all cl-representations of the same superclass object.

$\approx_{cl}$ must be (made) compatible with the functions of SP:

$$\forall\, u_1, \ldots, u_n : \quad t \approx_{cl} t' \quad \Rightarrow \quad f(t, u_1, \ldots, u_n) \approx_{SP} f(t', u_1, \ldots, u_n).$$

RESERVATION = ... `then`

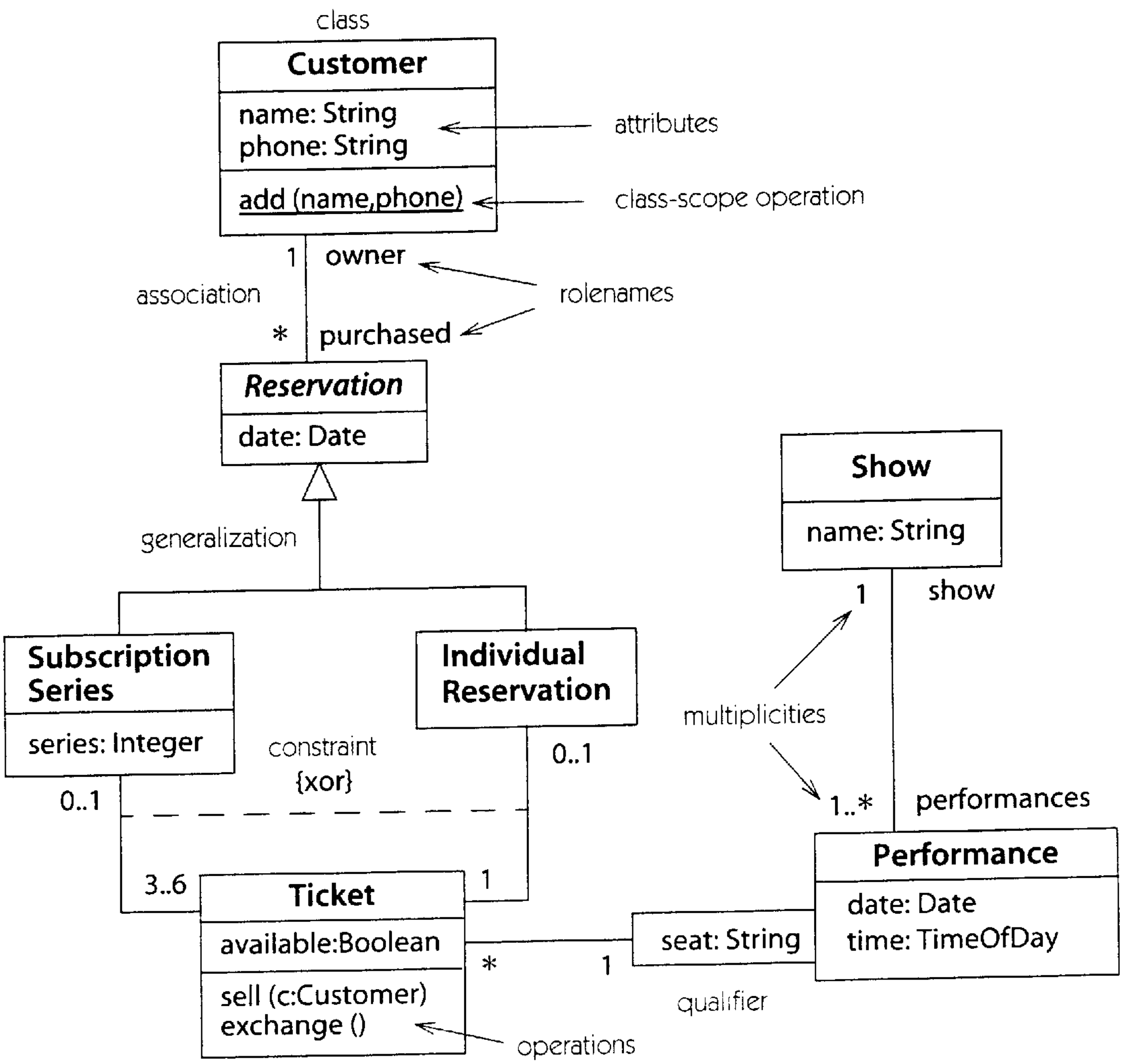
`dessorts` $Customer \quad Reservation \quad SS \quad IR \quad Ticket \quad Show \quad Performance$

$CustomId = String$

$TicketId = Date \times (int + 1) \times String \times TimeOfDay$

`destructs`
$name : Customer \rightarrow String$
$phone : Customer \rightarrow String$
$purchased : Customer \rightarrow Reservation^*$
$date : Reservation \rightarrow Date$
$owner : Reservation \rightarrow Customer$
$series : SS \rightarrow int$
$tickets : SS \rightarrow Ticket^*$
$ticket : IR \rightarrow Ticket$
$available : Ticket \rightarrow bool$
$ID : Ticket \rightarrow Date \times (int + 1) \times String \times TimeOfDay$
$subscriptionSeries : Ticket \rightarrow 1 + SS$
$individualReservation : Ticket \rightarrow 1 + IR$
$qperformance : Ticket \rightarrow String \times Performance$
$name : Show \rightarrow String$
$performances : Show \rightarrow Performance^*$
$date : Performance \rightarrow Date$
$time : Performance \rightarrow TimeOfDay$
$show : Performance \rightarrow Show$
$seat : String \times Performance \rightarrow String$
$ticket : String \times Performance \rightarrow Ticket$

`constructs`
$new : CustomId \times String \rightarrow Customer$
$new : TicketId \rightarrow Ticket$
$new : Date \times TimeOfDay \rightarrow Performance$
$new : int \rightarrow SS$
$new : \rightarrow IR$
$buy : Customer \times Ticket \rightarrow Customer$
$sell : Ticket \times Customer \rightarrow Ticket$
$exchange : Ticket \rightarrow Ticket$



from: J. Rumbaugh, I. Jacobson, G. Booch,
The Unified Modeling Language Reference Manual,
Addison-Wesley 1999

$$
\begin{array}{ll}
 & in^{Res}_{SS} : SS \to Reservation \\
 & in^{Res}_{IR} : IR \to Reservation \\
\texttt{vars} & c : Customer \quad cid : CustomId \quad t : Ticket \\
 & tid : TicketId \quad pho, sea : String \quad ss : SS \quad ir : IR \\
 & sh : Show \quad b : bool \quad i : int \quad dat : Date \\
 & tim : TimeOfDay \quad p : Performance \\
\texttt{Horn axioms} & name(new(cid, pho)) \equiv cid \\
 & phone(new(cid, pho)) \equiv pho \\
 & purchased(new(cid, pho)) \equiv \emptyset \\
 & name(buy(c, t)) \equiv name(c) \\
 & phone(buy(c, t)) \equiv phone(c) \\
 & purchased(buy(c, t)) \\
 & \quad \equiv (in^{Res}_{SS}, r_1, \ldots, r_n) \\
 & \qquad \Leftarrow subscriptionSeries(t) \equiv (ss) \wedge \\
 & \qquad\quad purchased(c) \equiv (r_1, \ldots, r_n) \\
 & purchased(buy(c, t)) \\
 & \quad \equiv (in^{Res}_{SS}, r_1, \ldots, r_n) \\
 & \qquad \Leftarrow individualReservation(t) \equiv (ir) \wedge \\
 & \qquad\quad purchased(c) \equiv (r_1, \ldots, r_n) \\
 & available(new(tid)) \equiv true \\
 & ID(new(tid)) \equiv tid \\
 & subscriptionSeries(new(dat, (i), sea, tim)) \\
 & \quad \equiv (new(i)) \\
 & subscriptionSeries(new(dat, (), sea, tim)) \equiv () \\
 & individualReservation(new(dat, (i), sea, tim)) \\
 & \quad \equiv () \\
 & individualReservation(new(dat, (), sea, tim)) \\
 & \quad \equiv (new) \\
 & qperformance(new(dat, (i), sea, tim)) \\
 & \quad \equiv (sea, new(date, time)) \\
 & available(sell(t, c)) \equiv false \\
 & ID(sell(t, c)) \equiv ID(t) \\
 & subscriptionSeries(sell(t, c)) \equiv subscriptionSeries(t)
\end{array}
$$

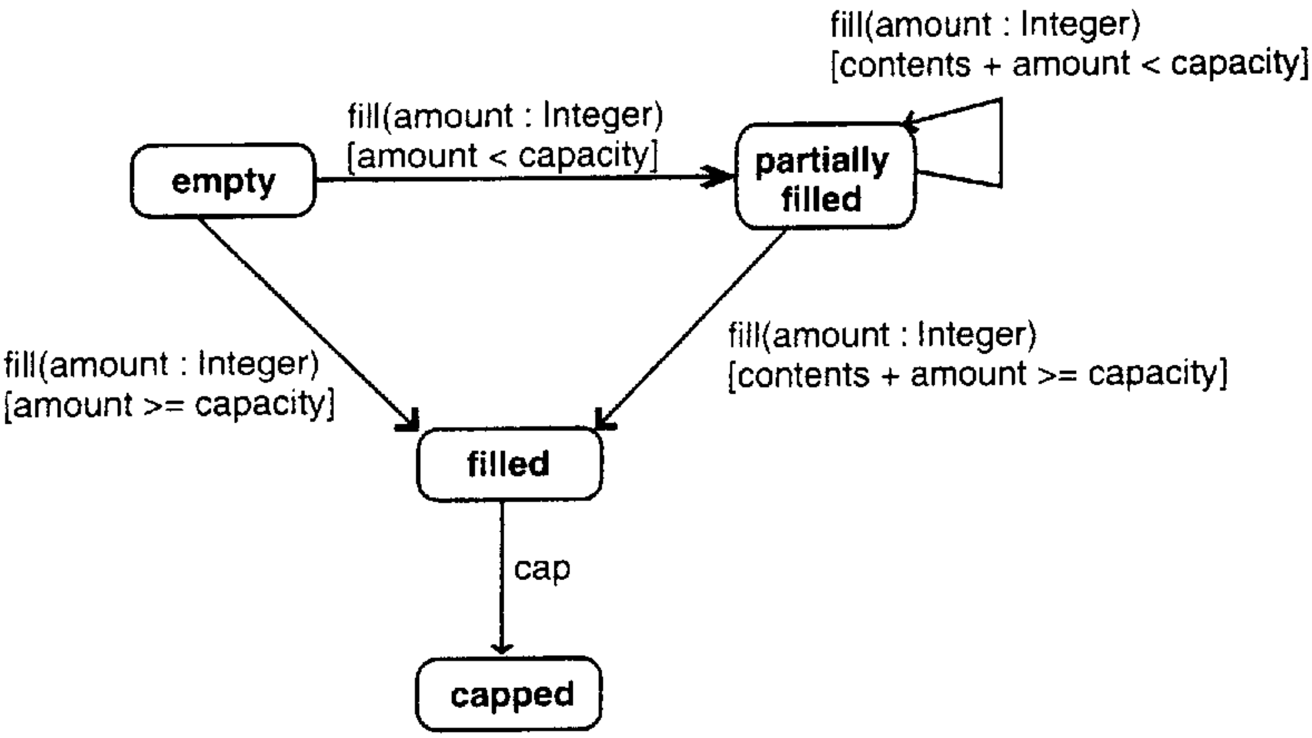
$$individualReservation(sell(t,c))$$
$$\equiv individualReservation(t)$$
$$qperformance(sell(t,c)) \equiv qperformance(t)$$
$$seat(sea,p) \equiv sea$$

`assertions` $3 \leq |mkset(tickets(ss))| \leq 6$

$$subscriptionSeries(t) \equiv () \ \dot{\vee}$$
$$individualReservation(t) \equiv ()$$
$$|mkset(performances(sh))| > 0$$

State machines

$$state_1 \xrightarrow{event(x)[guard(x)]/action(x)} state_2.$$



from: J.B. Warmer, A.G. Kleppe,
The Object Constraint Language, Addison-Wesley 1999

BOTTLESTATE = NAT `then`

`gensorts`	$Bottle$
`constructs`	$new : nat \to Bottle$
	$fill : Bottle \times nat \to Bottle$
	$cap : Bottle \to Bottle$
`destructs`	$capacity, contents : Bottle \to nat$
	$capped : Bottle$
`static` μ`-preds`	$empty, filled, partially\_filled : Bottle$
`vars`	$n : nat \quad b : Bottle$
`Horn axioms`	$capacity(new(n)) \equiv n$
	$contents(new(n)) \equiv 0$
	$capacity(fill(b, n)) \equiv capacity(b)$
	$contents(fill(b, n))$
	$\quad \equiv \quad min(contents(b) + n, capacity(b))$
	$capacity(cap(b)) \equiv capacity(b)$
	$contents(cap(b)) \equiv contents(b)$
	$capped(cap(b))$
	$empty(b) \quad \Leftarrow \quad contents(b) \equiv 0$
	$filled(b) \quad \Leftarrow \quad contents(b) \equiv capacity(b)$
	$partially\_filled(b)$
	$\quad \Leftarrow \quad 0 < contents(b) < capacity(b)$

☞ $empty$, $filled$ are $partially\_filled$ represent sets of attribute values,

☞ $capped$ denotes an object history.

Introducing events and transitions:

BOTTLETRANS $=$ **BOTTLESTATE** `then`

`vissorts` $event$

`constructs` $Fill: nat \rightarrow event$

$Cap: \rightarrow event$

`dynamic preds` $\_ \xrightarrow{-} \_ : Bottle \times event \times Bottle$

`Horn axioms` $b \xrightarrow{Fill(n)} fill(b,n) \quad \Leftarrow \quad empty(b)$

$b \xrightarrow{Fill(n)} fill(b,n) \quad \Leftarrow \quad partially\_filled(b)$

$b \xrightarrow{Cap} cap(b) \quad \Leftarrow \quad filled(b)$

BOTTLES $=$ **BOTTLETRANS** `and` **FINSET** `then`

`constructs` $New: nat \rightarrow event$

`dynamic preds` $\_ \xRightarrow{-} \_ : set(Bottle) \times event \times set(Bottle)$

`vars` $b, b' : Bottle \quad bs : set(Bottle) \quad e : event$

`Horn axioms` $bs \xRightarrow{New(n)} insert(new(n), bs)$

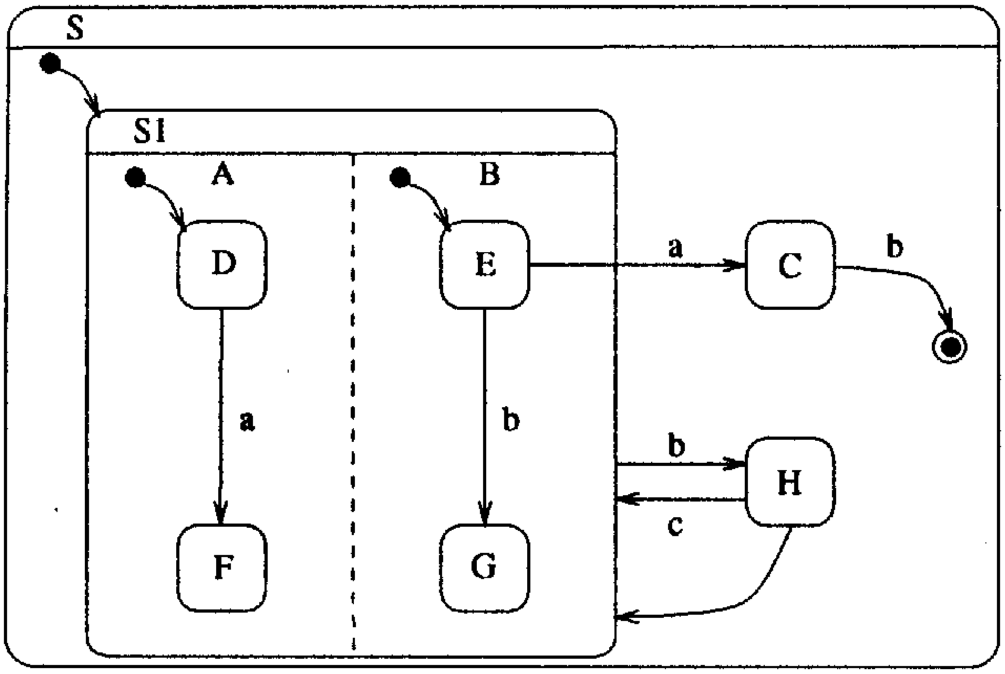
$$bs \xRightarrow{e} insert(b', remove(b, bs))$$
$$\Leftarrow \quad b \in bs \wedge b \xrightarrow{e} b'$$

MUTEX = NAT `and` **BOOL** `and` **FINSET** `then`

`vissorts`	$event \quad label$
`hidsorts`	$agent \quad resource$
	$environment = set(agent) \times resource$
`constructs`	$free, used\!:\rightarrow resource$
	$new\!:int \rightarrow agent$
	$access, release\!:agent \rightarrow agent$
	$New\!:nat \rightarrow event$
	$Request, Access, Release\!:\rightarrow event$
	$\_.\_\!:int \times event \rightarrow event$
	$\_\!:event \rightarrow label$
	$\_/\_\!:event \times event \rightarrow label$
`destructs`	$id\!:agent \rightarrow nat$
	$uses\!:agent \rightarrow bool$
`static preds`	$mutex : environment$
`dynamic preds`	$\_ \stackrel{\_}{\longrightarrow} \_ : agent \times label \times agent$
	$\_ \stackrel{\_}{\longrightarrow} \_ : resource \times label \times resource$
	$\_ \stackrel{\_}{\Longrightarrow} \_ : set(agent) \times label \times set(agent)$
	$\_ \Longrightarrow \_ : environment \times environment$
ν`-preds`	$\Box mutex : environment$
`vars`	$e, act : event \quad i : nat \quad a, a' : agent$
	$r, r' : resource \quad s, s' : set(agent)$
	$env, env' : environment$
`Horn axioms`	$id(new(i)) \equiv i$
	$id(access(a)) \equiv id(a)$
	$id(release(a)) \equiv id(a)$
	$uses(new(i)) \equiv false$
	$uses(access(a)) \equiv true$
	$uses(release(a)) \equiv false$

$$
\begin{array}{ll}
 & a \xrightarrow{Request} a \\
 & a \xrightarrow{Access} access(a) \\
 & a \xrightarrow{Release} release(a) \quad \Leftarrow \quad uses(a) \equiv true \\
 & free \xrightarrow{i.Request/i.Access} used \\
 & used \xrightarrow{Release} free \\
 & s \xRightarrow{New(i)} insert(new(i), s) \\
 & s \xRightarrow{i.e} insert(a', remove(a, s)) \\
 & \qquad \Leftarrow \quad a \in s \ \wedge \ a \xrightarrow{e} a' \ \wedge \ id(a) \equiv i \\
 & (s, r) \Longrightarrow (s', r') \quad \Leftarrow \quad s \xRightarrow{e} s' \ \wedge \ r \xrightarrow{e/act} r' \\
 & (s, r) \Longrightarrow (s', r') \quad \Leftarrow \quad r \xrightarrow{e/act} r' \ \wedge \ s \xRightarrow{act} s' \\
 & (s, r) \Longrightarrow (s', r') \quad \Leftarrow \quad s \xRightarrow{i.e} s' \ \wedge \ r \xrightarrow{e} r' \\
 & (s, r) \Longrightarrow (s', r) \quad \Leftarrow \quad s \xRightarrow{New(i)} s' \\
 & mutex(s, r) \quad \Leftarrow \quad |filter(uses, s)| \leq 1 \\
\texttt{co-Horn axioms} & \Box mutex(env) \quad \Rightarrow \quad mutex(env) \\
 & \Box mutex(env) \\
 & \qquad \Rightarrow \quad (env \Longrightarrow env' \quad \Rightarrow \quad \Box mutex(env')) \\
\texttt{theorem} & \Box mutex(\emptyset, free)
\end{array}
$$

Introducing composite states:



from: J. Lilius, I.P. Paltor,
The Semantics of UML State Machines,
TUCS Technical Report 273, Turku Centre for Computer Science 1999

ALPHABET

`vissorts`	$event$
`gensorts`	$outer \quad middle \quad A \quad B$
`constructs`	$a, b, c : \to event$
	$: \to event$
`objconstructs`	$S : middle \to outer$
	$S1 : A \times B \to middle$
	$C, H, final : \to middle$
	$D, F : \to A$
	$E, G : \to B$
`dynamic preds`	$\_ \xrightarrow{-} \_ : outer \times event \times outer$
`vars`	$x : A \quad y : B \quad z : middle$
`Horn axioms`	$S(S1(D, y)) \xrightarrow{a} S(S1(F, y))$
	$S(S1(x, E)) \xrightarrow{b} S(S1(x, G))$
	$S(S1(x, E)) \xrightarrow{a} S(C)$
	$S(C) \xrightarrow{b} S(final)$
	$S(S1(x, y)) \xrightarrow{b} S(H)$
	$S(H) \xrightarrow{c} S(S1(D, E))$
	$S(H) \longrightarrow S(S1(D, E))$

☞ A *sequential* composite state SS is a constructor whose domain sort agrees with the sort of the substates of SS.

☞ A *concurrent* composite state CS is a constructor whose domain sort is the *product* of sorts each of which represents a thread of CS.

UML models as logical theories

☞ *Combined translation of class diagrams, OCL constraints and state machines into coalgebraic STs.*
☞ In particular: binary associations and inheritance relationships are turned into destructors.

- Makes UML models amenable to prover-supported analysis and verification.
- Reveals inconsistencies and — often confusing — redundancies in UML concepts.

☞ *Model refinement via signature morphisms.*
☞ In particular: refinements of static to dynamic predicates and of structural to behavioral equalities.

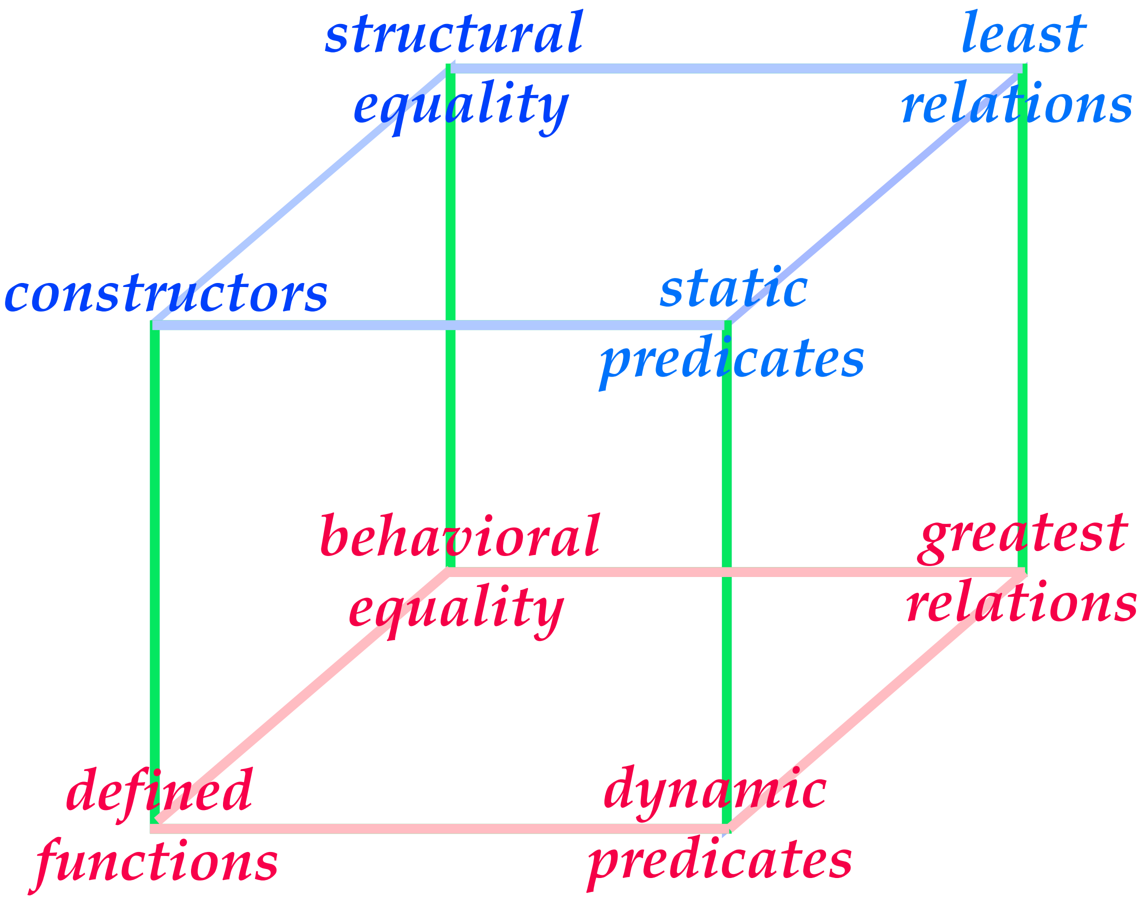
☞ *Future work*

- classification of proof obligations ("verification cases") for UML models
- comparison of ST models with relational ones
- design of a tailor-made *compile-time* prover complementing the existent *run-time* provers that check model *instances* against constraints
- translation of further UML concepts
- extension of UML by other concepts that are expressible in terms of STs (LTS, process calculi,...)

The Swinging Cube

Swinging types are an axiomatic design specifications of functions and relations,

- ➢ employing *many-sorted logic*,
- ➢ *Herbrand models* and *final coalgebras*.



Characteristics of STs

One-tiered approach: static and dynamic system aspects are treated within *many-sorted logic*.
Hierarchy: stepwise construction, stratified model.
Unique semantics in terms of *Herbrand models*
Each element of the Herbrand model denotes ➢ the static structure of an object or ➢ a set of attribute values or ➢ a composition of substates or ➢ the history of an object or ➢ the possible actions on an object or ➢ an "infinite" data structure or ➢ an element of an uncountable domain.
Each element of the Herbrand model is (1) a *term* consisting of *constructors* or (2) a *behavior* or interpretation of *contexts* that consist of *destructors* ($\approx$ *flowchart*) or (3) a *term* consisting of *constructors* and constants that denote *behaviors*.

How to specify subdomains: ➢ *Defined functions* generate sets of *normal forms* (= *constructor terms*). ➢ *Assertions* select subsets of *behaviors*.
Transition systems and *I/O relations* are *dynamic predicates*. ➢ Integration of *modal logic*.
There are *structural* and *behavioral* equalities.
Structural equality is *compatible* with functions and predicates, *inverse compatible* with constructors.
Behavioral equality is *compatible* with functions and *static predicates*, *zigzag compatible* with *dynamic predicates*, *inverse compatible* with *object constructors*.
Proof and computation rules rely on *functionality* (= existence of unique constructor terms w.r.t. structural equality) and the interpretation of μ-/ν-predicates as *least*/*greatest* solutions of *Horn*/*co-Horn* axioms. The rules include: ➢ *unfolding* of predicates and defined functions, ➢ *logical* and *non-logical simplifications*, ➢ *fixpoint induction* and *coinduction*, ➢ *contraction* and *expansion* upon theorems or constraints.

Basic Horn signature $\Sigma = (S, F, P)$

S	*sorts*
$F = CO \uplus DF$	*constructors*, *defined functions*
P	μ-*predicates*

For all $s \in S$,
μP implicitly contains a *structural equality* $\equiv: ss$.

Swinging signature $\Sigma = (S, F, P)$

Σ is a *swinging signature* if

- Σ is a basic Horn specification or
- $\Sigma = vis\Sigma \cup hid\Sigma$ where

$vis\Sigma = (visS, visF, visP)$ is a swinging signature (*visible subsignature of* Σ)

$hid\Sigma = (hidS, hidF, hidP)$ *hidden sorts/function symbols /predicates*

$hidF = CO \uplus DF$ *constructors*, *defined functions*

$f : w \to s \in CO \quad \Rightarrow \quad s \in hidS$,

CO includes a set of *object constructors.*

$hidP = \mu P \uplus \mu P' \cup \nu P$ *μ-predicates, ν-predicates*

$\mu P = statP \cup dynP$ *static/dynamic predicates*

Visible structural equalities are static *and* dynamic.

For all $s \in hidS$,

DF includes a set of *s-destructors* $f : sw \rightarrow s'$,

$statP$ includes a set of *s-destructors* $r : sw$,

$dynP$ includes a set of *s-transition predicates* $\delta : sws'$,

$dynP$ includes the *structural s-equality* $\equiv_s : ss$,

νP includes the *behavioral s-equality* $\sim_s : ss$.

If $vis\Sigma$ is basic Horn, then for all $s \in visS$,

νP includes the *behavioral s-equality* $\sim_s : ss$.

A predicate is *logical* if it is not a structural equality.

s-destructors and s-transition predicates are called *s-observers*.

Normal forms consist of constructors and variables.

$$\begin{aligned} NF_\Sigma(X) &= \text{set of } \Sigma\text{-normal forms} \\ NF_\Sigma &= \text{set of ground } \Sigma\text{-normal forms} \end{aligned}$$

Object normal forms consist of object constructors and variables.

Σ-formulas

- A *simple goal* is a finite conjunction of Σ-atoms.
- Let $Y \subseteq X$ and G, H be goals.
  $\forall Y(G \Rightarrow H)$ is a *universal goal*.
  $\exists Y G$ is an *existential goal*.
- Let f be a defined function and r be a logical predicate,
  $p = (f(t) \equiv u)$ resp. $p = r(t)$.
  Given a finite conjunction φ of universal goals,
  $p \Leftarrow \varphi$ is a *Horn clause for f resp. r*.
  $p \Leftarrow \varphi$ is *basic* if all universal goals of φ are unconditional.
- Given a simple goal G and a finite disjunction ψ of existential goals,
  $p \Rightarrow (G \Rightarrow \psi)$ is a *co-Horn clause for r*.
- *TRUE* is the empty conjunction.
  FALSE is the empty disjunction.

Co-Horn clauses $\overset{\text{negate}}{\longleftrightarrow}$ Horn clauses

Co-Horn axiom for the complement $\overline{r}$ of a predicate r:

$$\overline{r}(x,z) \;\Rightarrow\; (r(x,z) \;\Rightarrow\; \mathit{FALSE})$$

Co-Horn axiom for a predicate r:

$$r(x) \;\Rightarrow\; (G \;\Rightarrow\; (\exists X_1(G_1 \wedge r_1(t_1)) \vee \cdots \vee \exists X_n(G_n \wedge r_n(t_n))))$$

Horn axiom for the complement $\overline{r}$:

$$\overline{r}(x) \;\Leftarrow\; (G \wedge \forall X_1(G_1 \Rightarrow \overline{r_1}(t_1)) \wedge \cdots \wedge \forall X_n(G_n \Rightarrow \overline{r_n}(t_n)))$$

Co-Horn axioms for behavioral equality:

$x \sim_s y \;\Rightarrow\; x \equiv y$ for all *visible sorts* s

$x \sim_s y \;\Rightarrow\; f(x,z) \sim f(y,z)$ for all *destructors* $f : sw \to s'$

$x \sim_s y \;\Rightarrow\; (r(x,z) \;\Rightarrow\; r(y,z))$ for all *destructors* $r : sw$

$x \sim_s y \;\Rightarrow\; (\delta(x,z,x') \;\Rightarrow\; \exists y'(\delta(y,z,y') \wedge x' \sim y'))$
for all *trans. preds* $\delta : sws'$

Horn axioms for behavioral inequality:

$x \not\sim_s y \;\Leftarrow\; x \not\equiv y$ for all *visible sorts* s

$x \not\sim_s y \;\Leftarrow\; f(x,z) \not\sim f(y,z)$ for all *destructors* $f : sw \to s'$

$x \not\sim_s y \;\Leftarrow\; (r(x,z) \;\wedge\; \overline{r}(y,z))$ for all *destructors* $r : sw$

$x \not\sim_s y \;\Leftarrow\; (\delta(x,z,x') \;\wedge\; \forall y'(\delta(y,z,y') \Rightarrow x' \not\sim y'))$
for all *trans. preds* $\delta : sws'$

Compiling modal logic into predicate logic

Let x and y be states.

$$
\begin{array}{lcl}
comp(x \models p) & = & r_p(x) \quad \text{for all state sets } p \\
comp(x \models \varphi \wedge \psi) & = & comp(x \models \varphi) \wedge comp(x \models \psi) \\
comp(x \models \varphi \vee \psi) & = & comp(x \models \varphi) \vee comp(x \models \psi) \\
comp(x \models \langle . \rangle \varphi) & = & \exists y(x \rightarrow y \wedge comp(y \models \varphi)) \\
comp(x \models [.]\varphi) & = & \forall y(x \rightarrow y \Rightarrow comp(y \models \varphi)) \\
comp(x \models \mu p. \bigvee_{i=1}^{n} \varphi_i) & = & r_p(x) \\
 & & \text{where } r_p \text{ is specified by the axioms} \\
 & & r_p(x) \Leftarrow comp(x \models \varphi_i), \ 1 \leq i \leq n \\
comp(x \models \nu p. \bigwedge_{i=1}^{n} \varphi_i) & = & r_p(x) \\
 & & \text{where } r_p \text{ is specified by the axioms} \\
 & & r_p(x) \Rightarrow comp(x \models \varphi_i), \ 1 \leq i \leq n
\end{array}
$$

Poly-modal and weakly modal formulas

A formula is *poly-modal* if it is equivalent to a formula built up by the following rules:

- A atom $r(t)$ with $r \in statP \cup \nu P$ is poly-modal.
- If φ and ψ are poly-modal, then $\neg\varphi$ and $\varphi \wedge \psi$ are poly-modal.
- If φ is poly-modal, then for all atoms $\delta(t, x)$ with $\delta \in dynP$ and $x \in X \setminus var(t)$, $\exists x(\delta(t, x) \wedge \varphi)$ is poly-modal.

A formula is *weakly modal with output* $out(\varphi) \subseteq X$ if it is equivalent to a formula built up by the following rules:

- A poly-modal formula is weakly modal with output $\emptyset$.
- An atom $\delta(t, x)$ with $\delta \in dynP$ and $x \in X \setminus var(t)$ is weakly modal with output $\{x\}$.
- If φ and ψ are weakly modal with outputs Y resp. Z such that $Z \cap free(\varphi) = \emptyset$, then $\varphi \wedge \psi$ is weakly modal with output $Y \cup Z$.
- If φ is weakly modal with output Y, then for all $x \in X$, $\exists x\varphi$ is weakly modal with output $Y \setminus \{x\}$.

Modal, poly-modal, weakly modal formulas

A formula φ is *modal* if φ is equivalent to a formula built up by the following rules:

- A unary atom $r(t)$ with $r \in statP \cup \nu P$ is modal.
- If φ and ψ are modal, then $\neg\varphi$ and $\varphi \wedge \psi$ are modal.
- If φ is modal, then for all atoms $\delta(t, x)$ such that $\delta \in dynP$, t is unary and $x \in \mathit{free}(\varphi) \setminus var(t)$, $\exists x(\delta(t, x) \wedge \varphi)$ is modal.

A formula is *poly-modal* if it is equivalent to a formula built up by the following rules:

- A atom $r(t)$ with $r \in statP \cup \nu P$ is poly-modal.
- If φ and ψ are poly-modal, then $\neg\varphi$ and $\varphi \wedge \psi$ are poly-modal.
- If φ is poly-modal, then for all atoms $\delta(t, x)$ with $\delta \in dynP$ and $x \in X \setminus var(t)$, $\exists x(\delta(t, x) \wedge \varphi)$ is poly-modal.

A formula is *weakly modal with output* $out(\varphi) \subseteq X$ if it is equivalent to a formula built up by the following rules:

- A poly-modal formula is weakly modal with output $\emptyset$.
- An atom $\delta(t, x)$ with $\delta \in dynP$ and $x \in X \setminus var(t)$ is weakly modal with output $\{x\}$.
- If φ and ψ are weakly modal with outputs Y resp. Z such that $Z \cap free(\varphi) = \emptyset$, then $\varphi \wedge \psi$ is weakly modal with output $Y \cup Z$.
- If φ is weakly modal with output Y, then for all $x \in X$, $\exists x\varphi$ is weakly modal with output $Y \setminus \{x\}$.

Congruence axioms for Σ: EQ_Σ

$x \equiv x$

$y \equiv x \quad \Leftarrow \quad x \equiv y$

$f(x_1, \ldots, x_n) \equiv f(y_1, \ldots, y_n) \quad \Leftarrow \quad \bigwedge_{i=1}^{n} x_i \equiv y_i$

for all $f : s_1 \ldots s_n \to s \in F$

$r(x_1, \ldots, x_n) \quad \Leftarrow \quad \bigwedge_{i=1}^{n} x_i \equiv y_i \wedge r(y_1, \ldots, y_n)$

for all $r : s_1 \ldots s_n \in \mu P$

$r(x_1, \ldots, x_n) \quad \Rightarrow \quad (\bigwedge_{i=1}^{n} x_i \equiv y_i \;\Rightarrow\; r(y_1, \ldots, y_n))$

for all $r : s_1 \ldots s_n \in \nu P$

Basic Horn specification $SP = (\Sigma, AX)$

Σ basic Horn signature

AX basic Horn clauses for defined functions and logical predicates of Σ

(axioms of SP)

AX implicitly includes EQ_Σ.

For all $(f(t)\{\equiv u\} \Leftarrow \varphi) \in AX$,

(a) t is a tuple of normal forms,
$var(u) \subseteq var(t) \cup freevar(\varphi)$.

Swinging specification $SP = (\Sigma, AX)$

SP is a *swinging specification* or *swinging type* if

- SP is a basic Horn specification or
- Σ is a swinging signature,
  $vis\Sigma$ is the visible subsignature of Σ,
  $visSP = (vis\Sigma, visAX)$ is a swinging specification
  (visible subspecification of SP),
  $AX = visAX \uplus \mu AX \uplus \nu AX \uplus \mu AX'$
  (axioms of SP),
  μAX is a set of basic Horn clauses for μP and defined functions of Σ,
  νAX is a set of co-Horn clauses for νP,
  $\mu AX'$ is a set of Horn clauses for $\mu P'$
  such that

- AX implicitly includes EQ_Σ,
- for all $f \in DF \cup statP$, $(f(t)\{\equiv u\} \Leftarrow \varphi) \in \mu AX$, $\delta \in dynP$ and $(\delta(t,u) \Leftarrow \varphi) \in \mu AX$,
  (a) holds true,
  (b) φ is weakly modal, $var(t) \cap out(\varphi) = \emptyset$,
- for all $(p \Leftarrow \varphi) \in \mu AX$,
  (c) φ does not include predicates of νP or $\mu P'$,
- for all $(r(t) \Rightarrow (\varphi \Rightarrow \psi)) \in \nu AX$, (a) and (c) hold true, $\varphi \Rightarrow \psi$ is poly-modal and ψ does not include predicates of $\mu P'$,
- for all $(p \Leftarrow \varphi) \in \mu AX'$ and universal goals $\forall Y(G \Rightarrow H)$ of φ, G does not include predicates of $\mu P'$.

The axioms for behavioral equalities are called *behavior axioms* and read as follows:

$$x \sim_s y \;\Rightarrow\; x \equiv_s y$$

for all $s \in visS$ if $visSP$ is basic Horn,

$$x \sim_s y \;\Rightarrow\; f(x,z) \sim_{s'} f(y,z)$$

for all $s \in hidS$ and destructors $f : sw \to s'$,

$$x \sim_s y \;\Rightarrow\; r(x,z) \Leftrightarrow r(y,z)$$

for all $s \in hidS$ and destructors $r : sw$,

$$x \sim_s y \;\Rightarrow\; (\delta(x,z,x') \;\Rightarrow\; \exists\, y'(\delta(y,z,y') \wedge x' \sim_{s'} y')),$$

$$x \sim_s y \;\Rightarrow\; (\delta(y,z,y') \;\Rightarrow\; \exists\, x'(\delta(x,z,x') \wedge x' \sim_{s'} y'))$$

for all $s \in hidS$ and transition predicates $\delta : sws'$,

$$x \sim_s y \;\Rightarrow\; (x \equiv_s f(x') \;\Rightarrow\; \exists\, y'(y \equiv_s f(y') \wedge x' \sim_{s'} y')),$$

$$x \sim_s y \;\Rightarrow\; (y \equiv_s f(y') \;\Rightarrow\; \exists\, x'(x \equiv_s f(x') \wedge x' \sim_{s'} y'))$$

for all $s \in hidS$ and object constructors $f : w \to s$,

$$\kappa_i(x) \sim_{\coprod_{i \in I} s_i} \kappa_i(y) \;\Rightarrow\; x \sim_{s_i} y,$$

$$\kappa_i(x) \sim_{\coprod_{i \in I} s_i} \kappa_j(y) \;\Rightarrow\; FALSE$$

for all sorts s_i, $i \in I$, and $i \neq j \in I$

such that $s_k \in hidS$ for some $k \in I$,

$$(x_1, \ldots, x_n) \sim_{s_1 \ldots s_n} (y_1, \ldots, y_n) \;\Rightarrow\; x_i \sim_{s_i} y_i$$

for all sorts $s_1, \ldots, s_n$ and $1 \leq i \leq n$

such that $s_k \in hidS$ for some $1 \leq k \leq n$,

Behavioral equivalences

axioms for destructors *axioms for behavioral equality*

2-product equality

$\pi_1(x_1, x_2) \equiv x_1$ $\quad\quad$ $\vec{x} \sim \vec{y} \;\Rightarrow\; \pi_1(\vec{x}) \sim \pi_1(\vec{y})$

$\pi_2(x_1, x_2) \equiv x_2$ $\quad\quad$ $\vec{x} \sim \vec{y} \;\Rightarrow\; \pi_2(\vec{x}) \sim \pi_2(\vec{y})$

2-sum equality

$$x \sim x' \;\Rightarrow\; (x \equiv \kappa_1(y) \Rightarrow \exists y' : (x' \equiv \kappa_1(y') \wedge y \sim y'))$$

$$x \sim x' \;\Rightarrow\; (x \equiv \kappa_2(y) \Rightarrow \exists y' : (x' \equiv \kappa_2(y') \wedge y \sim y'))$$

strong equality

$$x \sim x' \;\Rightarrow\; (x \equiv \kappa_1(y) \Rightarrow \exists y' : (x' \equiv \kappa_1(y') \wedge y \sim y'))$$

list equality

$$L \sim L' \;\Rightarrow\; (L \equiv \kappa_i(\vec{x}) \Rightarrow \exists \vec{y}:(L' \equiv \kappa_i(\vec{y}) \wedge \vec{x} \sim \vec{y})) \quad (i \in \mathbb{N})$$

$\pi_i(x_1, \ldots, x_n) \equiv x_i$ $\quad\quad$ $(x_1, \ldots, x_n) \equiv (y_1, \ldots, y_n) \;\Rightarrow\; \pi_i(x) \sim \pi_i(x) \quad (1 \le i \le n)$

function equality $\quad$ $f \sim f' \;\Rightarrow\; apply(f, x) \sim apply(f', x)$

$apply(\lambda x.f(x, y), z) \equiv f(z, y)$

$apply(f \circ g, x) \equiv apply(f, apply(g, x))$

set equality $\quad$ $s \sim s' \;\Rightarrow\; in(x, s) \sim in(x, s')$

$in(x, s \cup s') \equiv (in(x, s) \; or \; in(x, s'))$

$in(x, compr(g)) \equiv g(x)$

bag equality $\quad$ $b \sim b' \;\Rightarrow\; card(x, b) \sim card(x, b')$

$card(x, b + b') \equiv card(x, b) + card(b', x)$

Non-alternating modal operators are μ- or ν-predicates:

`hidsorts`	$state$	
`static preds`	$enabled : state$	
	$\langle . \rangle r : state$	for all $r : state$
	$\Diamond r : state$	for all $r : state$
`dynamic preds`	$\_ \to \_ : state \times state$	
ν-`preds`	$disabled : state$	
	$[.]r : state$	for all $r : state$
	$\Box r : state$	for all $r : state$
`vars`	$s, s' : state$	
`Horn axioms`	axioms for $\to$	
	$enabled(s) \quad \Leftarrow \quad s \to s'$	
	$\langle . \rangle r(s) \quad \Leftarrow \quad s \to s' \wedge r(s')$	
	$\Diamond r(s) \quad \Leftarrow \quad r(s)$	
	$\Diamond r(s) \quad \Leftarrow \quad s \to s' \wedge \Diamond r(s')$	
`co-Horn axioms`	$disabled(s) \quad \Rightarrow \quad (s \to s' \Rightarrow FALSE)$	
	$[.]r(s) \quad \Rightarrow \quad (s \to s' \Rightarrow r(s'))$	
	$\Box r(s) \quad \Rightarrow \quad r(s)$	
	$\Box r(s) \quad \Rightarrow \quad (s \to s' \Rightarrow \Box r(s'))$	

Herbrand structure, complement, homomorphism

Let $\Sigma = (S, F, P)$ be a swinging signature and A, B be Σ-structures.

A is a *Herbrand structure* if

- for all $s \in S$, $A_s = T_{\Sigma,s}$,
- for all $f : w \to s \in F$ and $t \in T_{\Sigma,w}$, $f^A(t) = f(t)$.

A predicate $\overline{r} : w$ is the *complement* of a predicate $r : w$ *w.r.t. A* if $\overline{r}^A = A_w \setminus r^A$.

A *$\mu\nu\Sigma$-homomorphism* (*$\nu\mu\Sigma$-homomorphism*) $h : A \to B$ is an S-sorted function such that

- for all $f : w \to s \in F$, $h_s \circ f^A = f^B \circ h_w$,
- for all μ-predicates r, $h(r^A) \subseteq r^B$ ($r^B \subseteq h(r^A)$),
- for all ν-predicates r, $r^B \subseteq h(r^A)$ ($h(r^A) \subseteq r^B$).

Congruences and weak congruences

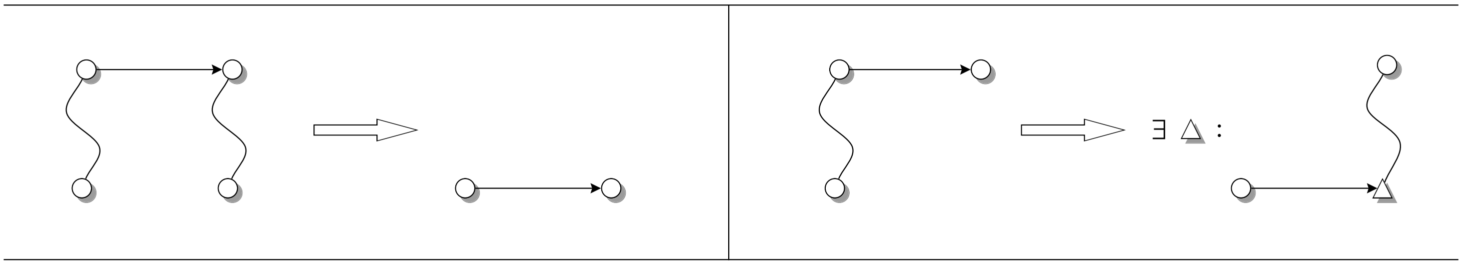
Let $\approx \subseteq A \times B$ be an S-sorted binary relation.

$\approx$ is *compatible* with $f : w \to s \in F$ resp. $r : w \in P$ if

$$\forall\, a \in A_w,\;\; b \in B_w : a \approx b \;\Rightarrow\; \begin{cases} f^A(a) \approx f^B(b) \text{ resp.} \\ a \in r^A \Leftrightarrow b \in r^B \end{cases}$$

$\approx$ is *zigzag compatible* with $\delta : ws \in P$ if

$$\forall\, (a, a') \in \delta^A : a \approx b \;\Rightarrow\; \exists\, (b, b') \in \delta^B : a' \approx b',$$
$$\forall\, (b, b') \in \delta^B : a \approx b \;\Rightarrow\; \exists\, (a, a') \in \delta^A : a' \approx b'.$$



Compatibility versus zigzag compatibility with $\longrightarrow$

$\approx$ is zigzag compatible with $\delta : ws$ iff $\approx$ is compatible with the non-deterministic function $f_\delta : w \to set(s)$ where

$$A_{set(s)} \;=\; \wp(A_s), \quad f_\delta^A(a) \;=\; \{a' \in A_s \mid (a, a') \in \delta^A\}$$

$$B \approx_{set(s)} C \quad\Longleftrightarrow\quad \begin{cases} \forall\, a \in B \;\exists\, b \in C : a \approx_s b, \\ \forall\, b \in C \;\exists\, a \in B : a \approx_s b. \end{cases}$$

A *Σ-congruence* is an equivalence relation that is compatible with all symbols of Σ.

A *weak Σ-congruence* is an equivalence relation that is compatible with all function symbols and static predicates of Σ and zigzag compatible with all dynamic predicates of Σ.

Quotient:
For all static predicates r,
$$r^{A/\approx} \quad = \quad \{[a] \mid a \in r^A\}.$$
For all dynamic predicates δ,
$$\delta^{A/\approx} \quad = \quad \{([a],[b]) \mid \exists\, b' \approx b : (a,b') \in \delta^A\}.$$

Hennessy-Milner Theorem

Let $\sim^A$ be a weak congruence. For all $b, c : X \to A$, $b \sim^A c$ iff for all poly-modal formulas φ,

$$A \models_b \varphi \quad \Longleftrightarrow \quad A \models_c \varphi.$$

A Σ-structure A is an *SP-model* if A satisfies AX and EQ_Σ.

If $\sim^A$ is a weak congruence, then $A/\!\sim^A$ is an SP-model.

$\approx$ is a *Σ-bisimulation* if

- for all *unary* terms $t(x)$, $a \approx b$ implies $t(a) \approx t(b)$,
- for all *unary* static atoms $p(x)$,
  $a \approx b$ implies $A \models p(a) \Leftrightarrow B \models p(b)$,
- for all dynamic atoms $p(x,y)$,
  $A \models p(a,a') \wedge a \approx b$ implies $\exists\, b' : (B \models p(b,b') \wedge a' \approx b')$,
- for all dynamic atoms $p(x,y)$,
  $B \models p(b,b') \wedge a' \approx b'$ implies $\exists\, a' : (A \models p(a,a') \wedge a \approx b)$.

A formula φ is *bisimulation invariant* in a class $\mathcal{C}$ of Σ-structures if for all $A, B \in \mathcal{C}$, bisimulations $\approx\, \subseteq A \times B$, $b : X \to A$ and $c : X \to B$,

$$b \approx c \quad \text{implies} \quad A \models_b \varphi \Leftrightarrow B \models_c \varphi.$$

Modal Invariance Theorem

Modal formulas are bisimulation invariant in all classes of Σ-structures.

Unary first-order formulas that are bisimulation invariant in some elementary class $\mathcal{C}$ of Σ-structures are modal in $\mathcal{C}$.

Deductive theory

The *deductive theory* of SP, $DTh(SP)$, is the least set F of formulas such that $TRUE \in F$ and, whenever the premises of one of the following inference rules are in F, the conclusion is also in F.

axiom rule	$\dfrac{TRUE}{\varphi} \Downarrow$	for all $\varphi \in AX \cup EQ_\Sigma$
instantiation	$\dfrac{\varphi}{\varphi\sigma} \Downarrow$	for all $\sigma : X \to T_\Sigma(X)$
modus ponens	$\dfrac{\{p \Leftarrow \varphi,\ \varphi\}}{p} \Downarrow$	
$\wedge$-*introduction*	$\dfrac{\{\varphi,\ \psi\}}{\varphi \wedge \psi} \Downarrow$	
$\forall$-*introduction*	$\dfrac{\{\varphi[t/x] \mid t \in T_\Sigma\}}{\forall x : \varphi} \Downarrow$	

Fixpoint theorems

Let U be a sorted set and $\Phi : \wp(U) \to \wp(U)$ be a monotone function with respect to sorted set inclusion.

Φ is *upward continuous* if for all increasing chains $B_0 \subseteq B_1 \subseteq B_2 \subseteq \ldots$ of subsets of U,

$$\Phi(\cup_{i\in\mathbb{N}} B_i) \quad \subseteq \quad \cup_{i\in\mathbb{N}} \Phi(B_i).$$

Φ is *downward continuous* if for all decreasing chains $B_0 \supseteq B_1 \supseteq B_2 \supseteq \ldots$ of subsets of U,

$$\cap_{i\in\mathbb{N}} \Phi(B_i) \quad \subseteq \quad \Phi(\cap_{i\in\mathbb{N}} B_i).$$

Kleene closures:

$$\Phi^\infty = \cup_{i\in\mathbb{N}} \Phi^i(\emptyset) \subseteq \mathit{lfp}(\Phi) \qquad \Phi_\infty = \cap_{i\in\mathbb{N}} \Phi^i(U) \supseteq \mathit{gfp}(\Phi)$$

Knaster-Tarski Theorem

$$\mathit{lfp}(\Phi) = \cap\{B \subseteq A \mid \Phi(B) \subseteq B\}$$

is the least fixpoint of Φ and a superset of Φ^∞.

$$\mathit{gfp}(\Phi) = \cup\{B \subseteq A \mid B \subseteq \Phi(B)\}$$

is the greatest fixpoint of Φ and a subset of Φ_∞.

Kleene's Theorem

If Φ upward continuous, then $\mathit{lfp}(\Phi) = \Phi^\infty$.

If Φ downward continuous, then $\mathit{gfp}(\Phi) = \Phi_\infty$.

Consequence operators

Let $SP = (\Sigma, AX) \subseteq SP' = (\Sigma', AX')$ *be swinging types,* $\Sigma'' = \Sigma' \setminus \Sigma$, $AX'' = AX' \setminus AX$, $A \in Struct_\Sigma$ *and* $\mathcal{C} = \{B \in Struct_{\Sigma'} \mid B|_\Sigma = A\}$.

The *AX''-consequence operator on A*, $\Phi{:}\mathcal{C} \to \mathcal{C}$, is defined as follows:

For all μ-predicates $r \in \Sigma''$ and $B \in \mathcal{C}$,

$$a \in r^{\Phi(B)} \iff_{def} \exists\ (r(t) \Leftarrow \varphi) \in AX'',\ b{:}X \to A:\\ a = b^*(t)\ \wedge\ B \models_b \varphi.$$

For all ν-predicates $r \in \Sigma''$ and $B \in \mathcal{C}$,

$$a \in r^{\Phi(B)} \iff_{def} \forall\ (r(t) \Rightarrow \varphi) \in AX'',\ b{:}X \to A:\\ a = b^*(t)\ \Rightarrow\ B \models_b \varphi.$$

Herbrand model $Her(SP)$

The *Herbrand SP-model*, $Her(SP)$, is the Herbrand Σ-structure that is defined as follows:

- If SP is basic Horn, then for all predicates $r : w \in \Sigma$,
  $r^{Her(SP)} =_{def} \{t \in T_{\Sigma,w} \mid r(t) \in DTh(SP)\}$.
- Otherwise let $visSP = (vis\Sigma, visAX)$ be the visible subspecification of SP, $SP' = (\Sigma', AX')$ be the union of the *basic Horn translation of visSP* and $(\mu P, \mu AX)$.
  Then for all predicates $r : w \in \Sigma'$,
  $r^{Her(SP)} =_{def} \{t \in T_{\Sigma',w} \mid r(t) \in DTh(SP')\}$,
  for all $r \in \nu P$, $r^{Her(SP)} =_{def} r^{gfp(\Phi)}$
  where Φ is the νAX-consequence operator on $Her(SP)|_{\Sigma'}$,
  for all $r \in \mu P'$, $r^{Her(SP)} =_{def} r^{lfp(\Psi)}$
  where Ψ is the $\mu AX'$-consequence operator on $Her(SP)|_{\Sigma' \cup \nu P}$.

φ is an *inductive theorem of SP* if $Her(SP) \models \varphi$

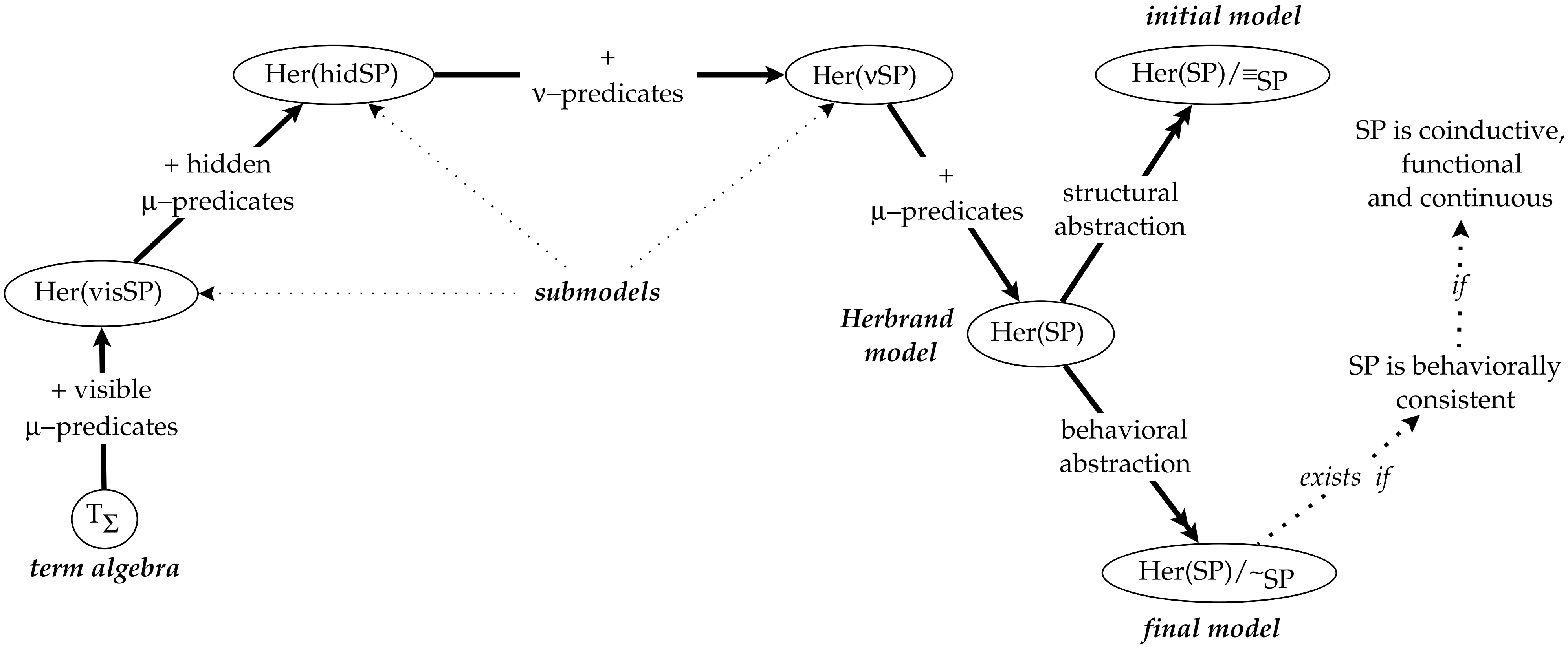
structural SP-equivalence: $\equiv_{SP} =_{def} \equiv^{Her(SP)}$

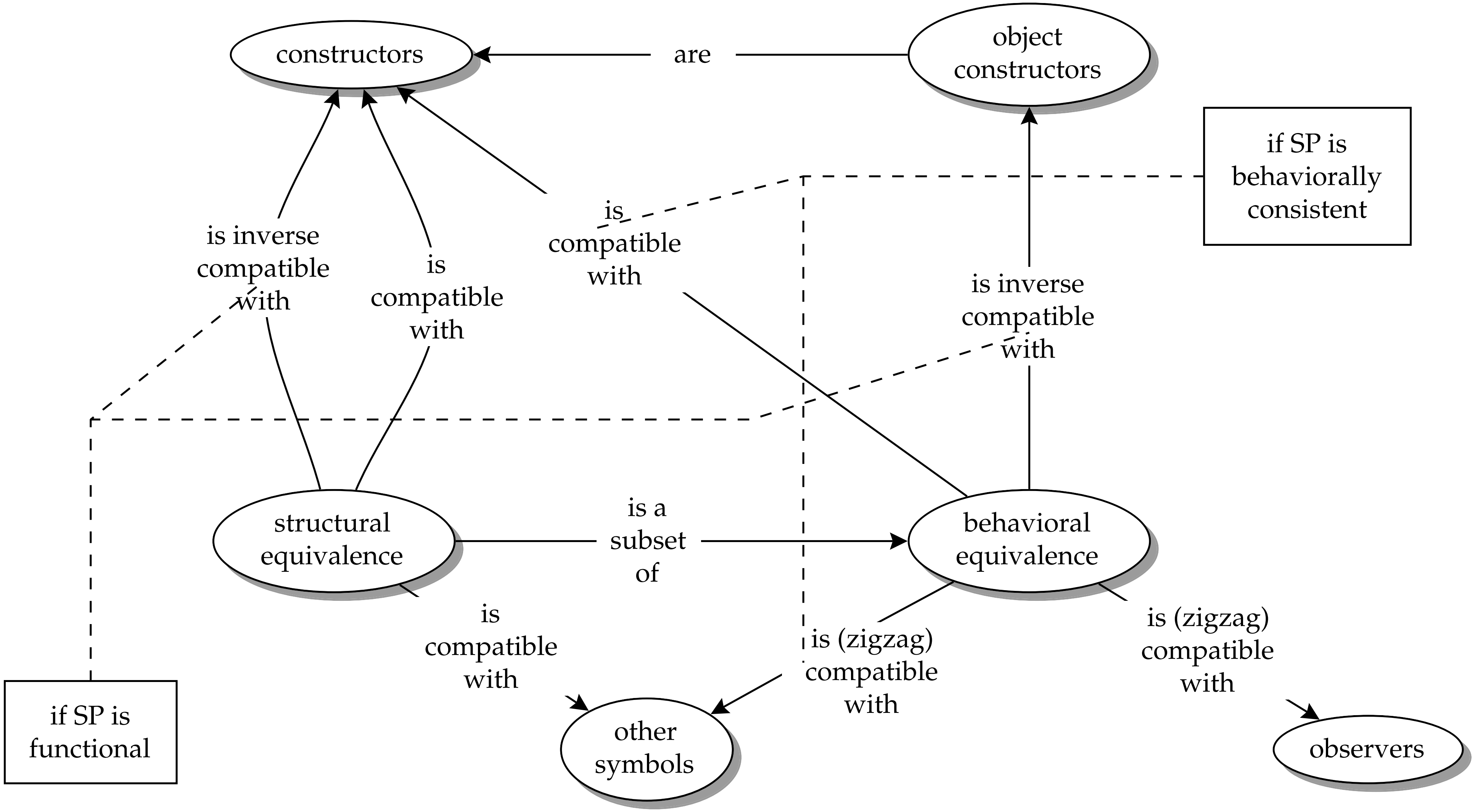
behavioral SP-equivalence: $\sim_{SP} =_{def} \sim^{Her(SP)}$

A sort $s \in \Sigma$ is *strongly visible* if $\equiv_{SP,s} = \sim_{SP,s}$.

Given $t \in T_\Sigma$, $u \in NF_\Sigma$ is a *normal form of* t if $t \equiv_{SP} u$.
SP is *complete* if each ground Σ-term has a normal form.
SP is *consistent* if structurally SP-equivalent ground normal forms are equal.
SP is *functional* if SP is complete and consistent. Then for all $t \in NF_\Sigma$, *nf(t)* denotes the unique normal form of t.
SP is *relational* if SP has no defined functions.
SP is *behaviorally consistent* if $\sim_{SP}$ is a weak Σ-congruence, i.e. $\sim_{SP}$ is compatible with $\Sigma \setminus dynP$ and zigzag compatible with $dynP$.
SP is *continuous* if the above consequence operators Φ and Ψ are downward resp. upward continuous.

$$\begin{array}{lll} \textit{initial SP-model:} & Ini(SP) =_{def} & Her(SP)/{\equiv_{SP}} \\ \textit{final SP-model:} & Fin(SP) =_{def} & Her(SP)/{\sim_{SP}} \\ & \text{if } SP \text{ is behaviorally consistent.} & \end{array}$$


initial model
Her(hidSP)
+
ν–predicates
Her(νSP)
Her(SP)/≡SP
+ hidden
μ–predicates
+
μ–predicates
structural
abstraction
SP is coinductive,
functional
and continuous
Her(visSP)
submodels
if
Herbrand
model
Her(SP)
SP is behaviorally
consistent
+ visible
μ–predicates
behavioral
abstraction
exists if
TΣ
term algebra
Her(SP)/~SP
final model


constructors
are
object constructors
if SP is behaviorally consistent
is inverse compatible with
is compatible with
is compatible with
is inverse compatible with
structural equivalence
is a subset of
behavioral equivalence
is compatible with
is (zigzag) compatible with
is (zigzag) compatible with
if SP is functional
other symbols
observers

Consequences of the functionality of SP

- *NF_Σ can be extended to a Σ-structure that is isomorphic to $Ini(SP)$.*
- *$\equiv_{SP}$ is inverse compatible with all constructors $c, d \in \Sigma$, i.e. $Her(SP)$ satisfies*

$$\begin{aligned} c(x) \equiv c(y) &\Rightarrow x \equiv y, \\ c(x) \equiv d(y) &\Rightarrow FALSE \qquad \text{if } c \neq d. \end{aligned}$$

- *$\sim_{SP}$ is inverse compatible with all object constructors $c, d \in \Sigma$, i.e. $Her(SP)$ satisfies*

$$\begin{aligned} c(x) \sim c(y) &\Rightarrow x \sim y, \\ c(x) \sim d(y) &\Rightarrow FALSE \text{ if } c \neq d. \end{aligned}$$

- *Axioms for inequalities:*

$$\begin{aligned} c(x_1,\dots,x_n) \not\equiv c(y_1,\dots,y_n) &\Leftarrow x_i \not\equiv y_i \\ &\text{for all constructors } c : s_1 \dots s_n \to s \\ &\text{and } 1 \le i \le n, \\ c(x) \not\equiv d(y) \ &\text{for all different constructors } c \text{ and } d. \end{aligned}$$

- *If* $visSP$ *is functional and continuous, then* SP *is monotone and consistent w.r.t.* $visSP$*, i.e. for all* $vis\Sigma$*-formulas* φ,

$$Her(SP) \models \varphi \iff Her(visSP) \models \varphi.$$

- *SP can be translated into an equivalent relational specification*

  Let $flat(\varphi)$ *be the flattened version of a formula* φ*, i.e. defined functions occur only at outermost positions of equations of* φ.

  Let $rel(flat(\varphi))$ *be obtained from* $flat(\varphi)$ *by replacing each equation* $f(t) \equiv u$ *with* $f \in DF$ *by the atom* $r_f(t,u)$ *where* r_f *is the graph of* f*. Then*

$$Her(SP) \models \varphi \iff Her(rel(flat(SP)) \models rel(flat(\varphi)).$$

Properties of the standard models

- *$Her(SP)$ is initial in $Mod(SP)$.*
- *$Ini(SP)$ is in $Mod_{\mu\nu}(SP)$.*
- *$Ini(SP)$ is initial in $Mod_{\equiv}(SP)$.*
- *For all first-order Σ-formulas φ and $\sigma : X \to T_\Sigma$,*

$$Her(SP) \models_\sigma \varphi \quad \Longleftrightarrow \quad Ini(SP) \models_{nat \circ \sigma} \varphi.$$

Suppose that SP is behaviorally consistent.

- *If SP is visible, then $Fin(SP)$ coincides with $Ini(SP)$.*
- *$Fin(SP)$ is in $Mod_{\mu\nu}(SP)$.*
- *$Fin(SP)$ is final in the class of reachable SP-models that interpret behavioral equality as a weak congruence.*
- *For all poly-modal formulas φ and $\sigma : X \to T_\Sigma$,*

$$Her(SP) \models_\sigma \varphi \quad \Longleftrightarrow \quad Fin(SP) \models_{nat \circ \sigma} \varphi.$$

Reductive theory, confluence

The *reductive theory* of SP, $RTh(SP)$, is the least set F of closed formulas such that $TRUE \in F$ and, whenever the conclusions of one of the following inference rules are in F, the premise is also in F.

rewriting $\quad \dfrac{p[t\sigma/x]}{p[u\sigma/x] \wedge \vartheta\sigma} \Uparrow \left\{ \begin{array}{l} \text{if } \varphi = (t \equiv u \Leftarrow \vartheta) \text{ is} \\ \text{an axiom, } \mathit{fresh}(\varphi)\sigma \subseteq NF_\Sigma \end{array} \right.$

resolution $\quad \dfrac{r(t\sigma)}{\vartheta\sigma} \Uparrow \left\{ \begin{array}{l} \text{if } r \neq \equiv,\ \varphi = (r(t) \Leftarrow \vartheta) \text{ is an} \\ \text{axiom, } \mathit{fresh}(\varphi)\sigma \subseteq NF_\Sigma \end{array} \right.$

reflection $\quad \dfrac{t \equiv t}{TRUE} \Uparrow$

$\wedge$-elimination $\quad \dfrac{\varphi \wedge \psi}{\{\varphi,\ \psi\}} \Uparrow$

$\forall$-elimination $\quad \dfrac{\forall x : \varphi}{\{\varphi[t/x] \mid t \in T_\Sigma\}} \Uparrow$

SP is *confluent* if $RTh(SP)^+ \subseteq RTh(SP)$.

<u>*Church-Rosser Theorem*</u>

A complete specification SP is functional iff SP is confluent iff all closed unconditional universal goals of $DTh(SP)$ are in $RTh(SP)$.

Criteria for functionality

A swinging specification SP is *terminating* if

- all SP-reduced terms are normal forms,
- there is a well-founded and transitive relation $>$ on ground terms and closed formulas such that
  - for all axioms $C = (r(t) \Leftarrow \varphi)$ and $\sigma : X \to T_\Sigma$,

    $$G\sigma \in RTh(SP) \wedge fresh(C)\sigma \subseteq NF_\Sigma \;\Rightarrow\; r(t\sigma) > G\sigma,$$

  - for all axioms $C = (t \equiv u \Leftarrow \varphi)$, terms and atoms $v(x)$ and $\sigma : X \to T_\Sigma$,

    $$G\sigma \in RTh(SP) \wedge fresh(C)\sigma \subseteq NF_\Sigma \;\Rightarrow\; v(t\sigma) > \{v(u\sigma), G\sigma\},$$

  - $t > u$ for all $t \in T_\Sigma$ and proper subterms u of t.

SP is *weakly orthogonal* if for all *SP-overlays* $t \equiv t' \Leftarrow \varphi$, $t = t'$ or $\neg\varphi$ is an inductive theorem of SP.

- *Terminating specifications are complete.*
- *Relational specifications are weakly orthogonal.*
- *Terminating and weakly orthogonal specifications are functional.*

A simple Horn clause

$$f(t)\{\equiv u\} \quad \Leftarrow \quad t_1 \equiv u_1 \wedge \cdots \wedge t_n \equiv u_n \wedge \varphi$$

is *deterministic* if for all $1 \leq i \leq n$,

- u_i is a normal form,
- $var(t_i) \subseteq V_{i-1}$, $var(u_i) \cap V_{i-1} = \emptyset$,
- $var(u) \subseteq V_n$

where $V_0 = var(t)$ and $V_i = V_{i-1} \cup var(u_i)$.

Hierarchical confluence criterion

Let $SP' = (\Sigma', AX') \subseteq SP = (\Sigma, AX)$ be swinging specifications. *SP is confluent if SP is terminating, SP' is confluent,*

- *all constructors of $(SP \setminus SP')$-sorts are in $\Sigma \setminus \Sigma'$,*
- *all axioms of $SP \setminus SP'$ are axioms for $(\Sigma \setminus \Sigma')-$symbols,*
- *for all conditional equations $\varphi, \psi \in AX \setminus AX'$,*
  - *$\varphi = \psi$ and φ is deterministic or*
  - *each overlay $p \Leftarrow \vartheta$ induced by φ and ψ is an inductive theorem of SP' or some atom of ϑ^+ has no ground instance in $RTh(SP)$.*

Image finite specification

$$p(s) \;\Rightarrow\; \exists s'(s \to s' \wedge p(s'))$$

$p(s)$ holds true iff some path $s \to s_1 \to s_2 \ldots$ is infinite,

$$q(s) \;\Leftarrow\; \forall s'(s \to s' \Rightarrow q(s'))$$

$q(s)$ holds true iff all paths $s \to s_1 \to s_2 \ldots$ are finite.

If $\to^{Her(SP)}$ is finitely branching, then the specification of p and q is continuous.

Axioms expressing that a stream s is fair, i.e. s satisfies g for infinitely many elements of s:

$$fair(g,s) \;\Rightarrow\; \exists n, s' : \begin{array}{l} \overbrace{nthtail(n,s) \equiv s' \wedge g(head(s')) \equiv true}^{G} \\ \wedge\ \underbrace{fair(tail(s'))}_{H} \end{array}$$

Since there are streams t such that $G[t/s]$ has infinitely many normal form solutions, $\exists n, s' : (G \wedge H)$ is not image finite.

$$fair(g,s) \;\Rightarrow\; \exists n, s' : \begin{array}{l} \overbrace{G \wedge forall(not \circ g, firstn(n,s))}^{G'} \\ \wedge\ \underbrace{fair(tail(s'))}_{H} \end{array}$$

Since for all streams t, $G'[t/s]$ has at most one normal form solution, $\exists n, s' : (G' \wedge H)$ is image finite.

Given a Σ-goal G, let

$$\mathcal{S}(G) \quad =_{def} \quad \{\tau : var(G) \to NF_\Sigma \mid Her(SP) \models G\tau\}.$$

Given $Y \subseteq X$, G is *Y-image finite* if for all $\sigma : X \to NF_\Sigma$, $\mathcal{S}(G\sigma_{X\setminus Y})$ is finite.

An existential $\nu\Sigma$-goal $\exists Y\varphi$ is *image finite* if φ is a $hid\Sigma$-goal or φ contains a nonempty Y-image finite $hid\Sigma$-goal.

A universal Σ-goal $\forall Y(G \Rightarrow H)$ is *image finite* if G and H are $\nu\Sigma$-goals or G is a Y-image finite $\nu\Sigma$-goal.

SP is *image finite* if

- all axioms $p \Rightarrow \varphi$ of $hidSP$ are Horn clauses,
- for all axioms $p \Rightarrow (G \Rightarrow \varphi)$ of $\nu SP \setminus hidSP$, φ consists of image finite existential goals,
- for all axioms $p \Leftarrow \varphi$ of $SP \setminus \nu SP$, φ consists of image finite universal goals.

If SP is complete and image finite, then SP is continuous.

Coinductive specification

Let t be a hidden term.

An atom $\delta(t, a, u)$ is *observing* if

- δ is a transition predicate or
- $\delta(t, a, u) = (f(t, a) \equiv u)$ and f is a destructor or
- $\delta(t, a, u) = r(t, a)$ and r is a destructor.

A Horn clause $\delta(t,a,u) \Leftarrow \varphi$ is *coinductive* if

$\delta(t,a,u)$ is not observing and *t is object normal* or

$\delta(t,a,u)$ is *observing*,

$\varphi \;\; = \;\; G \wedge \delta_1(t_1,a_1,u_1) \wedge G_1 \wedge \cdots \wedge \delta_n(t_n,a_n,u_n) \wedge G_n$,

- *t is object normal* or $t = c(t')$ for a constructor c and an *object normal form t'*, a is object normal, G is weakly modal and not observing, $var(u) \subseteq V_n$ and $out(G) \cap var(t,a) = \emptyset$,
- for all $1 \le i \le n$, $\delta_i(t_i,a_i,u_i)$ is *observing*, (t_i,a_i) is normal, *u_i is object normal*, G_i is weakly modal and not observing, $var(t_i) \subseteq V_{i-1}$, $hidvar(a_i) \subseteq var(a)$ and $(var(u_i) \cup out(G_i)) \cap (V_{i-1} \cup var(a_i,u_i)) = \emptyset$

where $\begin{cases} V_0 = var(t,a,G), \\ V_i = V_{i-1} \cup var(a_i,u_i,G_i). \end{cases}$

A co-Horn clause $r(t) \Rightarrow \varphi$ is *coinductive* if t is object normal.

SP is *coinductive* if

- for all axioms φ of $SP \setminus visSP$,
  φ is coinductive or an axiom for a non-observer f such that $\sim_{SP}$ is (zigzag) compatible with f,
- for all axioms $p \Leftarrow \varphi$ for observers and all non-observers f occurring in φ, the axioms for f are coinductive and do not contain observers.

A functional, continuous and coinductive specification is behaviorally consistent.

Behavioral equivalence is compatible with constructors defined by SOS rules

$$(*) \qquad \frac{t_1(x) \longrightarrow x_1,\ \ t_2(x, x_1) \longrightarrow x_2}{c(x) \longrightarrow t_3(x, x_1, x_2)}$$

The *constructor closure* $\approx$ of $\sim$:

(1) $\quad t \sim t' \quad \Rightarrow \quad t \approx t'$

(2) $\quad t \approx t' \quad \Rightarrow \quad c(t) \approx c(t')$

$\approx$ satisfies the behavior axioms $\quad \Rightarrow \quad \approx \subseteq \sim \quad \overset{(1)}{\Rightarrow} \quad \approx = \sim$

$\Rightarrow \quad \sim$ *is compatible with* c.

$\approx$ satisfies the behavior axioms:

Let $t' \approx t \longrightarrow v$. Get v' such that $t' \longrightarrow v' \approx v$.

Case (1): $t \sim t'$.

Then $t' \longrightarrow v' \sim v$ and thus $t' \longrightarrow v' \approx v$. ✔

Case (2): $t = c(u),\ t' = c(u'),\ u \approx u'$. By $(*)$,

$v = t_3(u, u_1, u_2),\ t_1(u) \longrightarrow u_1,\ t_2(u, u_1) \longrightarrow u_2$.

By (2), $t_1(u') \approx t_1(u) \longrightarrow u_1$.

By induction hypothesis, $t_1(u') \longrightarrow u_1' \approx u_1$.

By (2), $t_2(u', u_1') \approx t_2(u, u_1) \longrightarrow u_2$.

By induction hypothesis, $t_2(u', u_1') \longrightarrow u_2' \approx u_2$.

By (2), $t_3(u', u_1', u_2') \approx t_3(u, u_1, u_2)$.

By $(*)$, $t' = c(u') \longrightarrow t_3(u', u_1', u_2') \approx v$. ✔

Structure of a “natural” verifier

A proof of φ is an analytic tableau of the following form:

$$\varphi = \varphi_1 \quad \vdash \quad \varphi_2 \quad \vdash \quad \ldots \quad \vdash \quad \varphi_n = \psi$$

The applied rules are sound w.r.t. the Herbrand SP-model:

$$\varphi = \varphi_1 \quad \Leftarrow \quad \varphi_2 \quad \Leftarrow \quad \ldots \quad \Leftarrow \quad \varphi_n = \psi$$

Special cases:

- ψ is a constraint under which ϕ holds,
- $\psi = (x_1 \equiv t_1 \wedge \cdots \wedge x_n \equiv t_n)$ is a solution of ϕ,
- $\psi = TRUE$.

$\varphi_1, \ldots, \varphi_n$ are not normalized. Only at the beginning, negation symbols must be shifted inwards so that $\neg r(t)$ becomes $\overline{r}(t)$. Axioms for $\overline{r}$ can be derived from those for r.

Rules are of three kinds:

- An *expansion* $\frac{\varphi}{\psi}\Uparrow$ is applicable to $C[\varphi]$
  if φ is a *positive* subformula of $C[\varphi]$:

$$polarity(position(\varphi), C[\varphi]) = true \quad \Longrightarrow \quad \frac{C[\varphi]}{C[\psi]}\Uparrow$$

- A *contraction* $\dfrac{\varphi}{\psi}\Downarrow$ is applicable to $C[\varphi]$
  if φ is a *negative* subformula of $C[\varphi]$:

$$polarity(position(\varphi), C[\varphi]) = false \quad \Longrightarrow \quad \frac{C[\varphi]}{C[\psi]}\Uparrow$$

- A *simplification* $\dfrac{\varphi}{\psi}\Updownarrow$ is applicable to any $C[\varphi]$:

$$\frac{C[\varphi]}{C[\psi]}\Updownarrow$$

$$\begin{aligned}
polarity(w, \psi) &= f(w, \psi, true) \\
f(0w, \varphi \Rightarrow \psi, b) &= f(w, \varphi, not(b)) \\
f(1w, \varphi \Rightarrow \psi, b) &= f(w, \psi, b) \\
f(0w, \varphi \overset{\wedge}{\vee} \psi, b) &= f(w, \varphi, b) \\
f(1w, \varphi \overset{\wedge}{\vee} \psi, b) &= f(w, \psi, b) \\
f(0w, \overset{\forall}{\exists} x\varphi), b) &= f(w, \varphi, b) \\
f(\varepsilon, \varphi, b) &= b
\end{aligned}$$

Proof goals

- **Pre/post conditions**

$$f(x) \equiv y \quad \Rightarrow \quad (pre(x) \;\Rightarrow\; post(x, y))$$

Apply *fixpoint induction* on f.

Examples:

$part(L, P) \quad \Rightarrow \quad L \equiv flatten(P)$
Apply *fixpoint induction* on $part$.

$sort(L) \equiv L' \quad \Rightarrow \quad sorted(L') \wedge L \sim L'$
Apply *fixpoint induction* on the graph of $sort$.

- **Function invariants**

$$f(x) \equiv y \quad \Rightarrow \quad (inv(x) \;\Rightarrow\; inv(y))$$

Apply *fixpoint induction* on f.

Example:

$sorted(L) \wedge sorted(L') \quad \Rightarrow \quad sorted(merge(L, L'))$
Variable introduction:
$merge(L, L') \equiv L'' \;\Rightarrow\; ((sorted(L) \wedge sorted(L')) \;\Rightarrow\; sorted(L''))$

Apply *fixpoint induction* on the graph of $merge$:
$\exists\, q : \; q(L, L', L'') \;\Rightarrow\; ((sorted(L) \wedge sorted(L')) \Rightarrow sorted(L'')) \wedge$
$\quad \bigwedge_{\varphi \in AX(merge)} \varphi[q/\lambda(L, L', L'').merge(L, L') \equiv L'']$

Define q:

$$q(L, L', L'') \quad \Leftarrow \quad (sorted(L) \wedge sorted(L')) \Rightarrow sorted(L'')$$

Unfold q.

- **Transition invariants**

$$\psi \quad \Rightarrow \quad (\Box inv)(init)$$

Apply *coinduction* on $\Box inv$.

Example:

$\Box mutex(\emptyset, free)$

Apply *coinduction* on $\Box mutex$:

$\exists\ q :\ q(\emptyset, free) \wedge$

$\qquad (q(s) \Rightarrow mutex(s)) \ \wedge\ ((s \rightarrow s' \wedge q(s)) \Rightarrow q(s'))$

Define q:

$$\begin{array}{rcl} q(s, free) & \Leftarrow & |filter(uses, s)| \equiv 0 \\ q(s, used) & \Leftarrow & |filter(uses, s)| \leq 1 \end{array}$$

Unfold q and $mutex$.

- **Prove behavioral equations**

$$\psi \quad \Rightarrow \quad t(x) \sim u(y)$$

Apply *coinduction* on $\sim$.

Example:

$sort(L) \equiv L' \quad \Rightarrow \quad L \sim L' \qquad\qquad (1)$

Apply *coinduction* on $\sim$:
$\exists\ q:\ \ sort(L) \equiv L' \Rightarrow q(L,L') \wedge$
$\qquad q(L,L') \Rightarrow card(L)(x) \equiv card(L')(x)$

Define q:

$$q(L,L') \Leftarrow sort(L) \equiv L'$$

Unfold q:
$sort(L) \equiv L' \Rightarrow card(L)(x) \equiv card(L')(x)$
The same subgoal is achieved if the occurrence of $\sim$ in (1) is unfolded because $\sim$ is *derived*, i.e. specified *non-recursively*:

$$L \sim L' \Rightarrow card(L)(x) \equiv card(L')(x)$$

Apply *fixpoint induction* on $sort$.

Example:

$$finite(s) \vee fair(g \circ f, s) \Rightarrow filter(g, map(f, s)) \sim map(f, filter(g \circ f, s))$$

Apply *coinduction* on $\sim$:

$$\exists\, q : \; finite(s) \vee fair(g \circ f, s) \Rightarrow q(filter(g, map(f, s)), map(f, filter(g \circ f, s)) \wedge$$
$$(q(s_1, s_2) \wedge s_1 \xrightarrow{x} t_1) \Rightarrow \exists t_2 : (s_2 \xrightarrow{x} t_2 \wedge t_1 \sim t_2) \wedge$$
$$(q(s_1, s_2) \wedge s_2 \xrightarrow{x} t_2) \Rightarrow \exists t_1 : (s_1 \xrightarrow{x} t_1 \wedge t_1 \sim t_2)$$

Define q:

$$finite(s) \vee fair(g \circ f, s) \Rightarrow q(filter(g, map(f, s)), map(f, filter(g \circ f, s))$$

Unfold q.

The proof also employs *inductive contraction* along the following induction ordering $\gg$:

$$(g,f,s) \gg (g,f,t) \quad \Leftarrow \quad length(s,m) \wedge length(t,n) \wedge m > n$$
$$(g,f,s) \gg (g,f,t) \quad \Leftarrow \quad first(g \circ f, s, m) \wedge first(g \circ f, t, n) \wedge m > n$$

Axioms for $length$ and $first$:

$$length(s,0) \quad \Leftarrow \quad isnil(s)$$
$$length(s,n+1) \quad \Leftarrow \quad s \xrightarrow{x} t \wedge length(t,n)$$
$$first(g,s,0) \quad \Leftarrow \quad s \xrightarrow{x} t \wedge g(x) \equiv true$$
$$first(g,s,n+1) \quad \Leftarrow \quad s \xrightarrow{x} t \wedge g(x) \equiv false \wedge first(g,t,n)$$

- **Prove a property of another ν-predicate r**

$$\psi \quad \Rightarrow \quad r(t(x))$$

Apply *coinduction* on r.

Example:

$fair(\lambda x.eq(x,0), blink)$

Apply *coinduction* on $fair$:
$\exists\, q :\; q(\lambda x.eq(x,0), blink) \,\wedge\, (q(f,s) \Rightarrow (exists(f,s) \wedge q(f, tail(s))))$

Define q:

$$q(\lambda x.eq(x,0), blink)$$
$$q(\lambda x.eq(x,0), 1\&blink)$$

Unfold q.

- **Compute solutions of a μ-predicate r**

$$
\begin{array}{rcl}
r(x) & \stackrel{unfolding}{\vdash} & \bigvee_{i=1}^{n}(x \equiv t_i \wedge \varphi_i) \\
 & \vdash & \bigvee_{i=1}^{k}(x \equiv t_i \wedge \mathit{TRUE}) \vee \bigvee_{i=k+1}^{n}(x \equiv t_i \wedge \mathit{FALSE}) \\
 & \vdash & \bigvee_{i=1}^{k} x \equiv t_i
\end{array}
$$

Example:

$$
\begin{array}{ll}
 & r(n) \\
\vdash & as \stackrel{open(x);\ x.deposit(100);\ x.withdraw(50);\ x.bal?(n)}{\Longrightarrow} bs \\
\vdash & x \notin as \wedge n \equiv 50 \wedge \ldots
\end{array}
$$

- **Compute dissolutions of a ν-predicate r**

$$
\begin{array}{rcl}
r(x) & \stackrel{unfolding}{\vdash} & \bigwedge_{i=1}^{n}(x \equiv t_i \Rightarrow \psi_i) \\
 & \vdash & \bigwedge_{i=1}^{k}(x \equiv t_i \Rightarrow \mathit{TRUE}) \wedge \bigwedge_{i=k+1}^{n}(x \equiv t_i \Rightarrow \mathit{FALSE}) \\
 & \vdash & \bigwedge_{i=k+1}^{n} x \not\equiv t_i
\end{array}
$$

- **Derive a program for f**

<u>*Example:*</u>

$$f(x) \equiv g(h(x), h'(x), x)$$

Unfold g, h, h'

$$\vdash \quad \bigvee_{i=1}^{n}(x \equiv t_i \wedge f(t_i) \equiv u_i[g, h, h'])$$

Apply inductive expansion (here the same as *folding* of f)

$$\vdash \quad \bigvee_{i=1}^{n}(x \equiv t_i \wedge f(t_i) \equiv v_i[f])$$

Unfold f with the "derived program" $f(t_1) \equiv v_1, \ldots, f(t_n) \equiv v_n$

$$\vdash \quad \bigvee_{i=1}^{n} x \equiv t_i$$

Apply expansion with the lemma $x \equiv t_1 \vee \cdots \vee x \equiv t_n$

$$\vdash \quad \mathit{TRUE}$$

Proof and computation rules

unfolding of a μ-predicate r

$$\frac{r(t)}{\bigvee_{i=1}^{n} \exists Z_i : (t \equiv t_i \wedge \varphi_i)} \Updownarrow$$

where $r(t_1) \Leftarrow \varphi_1, \dots, r(t_n) \Leftarrow \varphi_n$ are the axioms for r
and $Z_i = var(t_i, \varphi_i)$

unfolding of a ν-predicate r

$$\frac{r(t)}{\bigwedge_{i=1}^{n} \forall Z_i : (t \equiv t_i \Rightarrow \varphi_i)} \Updownarrow$$

where $r(t_1) \Rightarrow \varphi_1, \dots, r(t_n) \Rightarrow \varphi_n$ are the axioms for r
and $Z_i = var(t_i, \varphi_i)$

unfolding of a defined function f

$$\frac{\varphi(f(t))}{\bigvee_{i=1}^{n} \exists Z_i : (t \equiv t_i \wedge \varphi(u_i) \wedge \varphi_i)} \Updownarrow \quad \textit{if SP is functional}$$

where $f(t_1) \equiv u_1 \Leftarrow \varphi_1, \dots, f(t_n) \equiv u_n \Leftarrow \varphi_n$ are the axioms for f
and $Z_i = var(t_i, u_i, \varphi_i)$

unfolding of a cofunction f

$$\frac{\varphi(d(f(t), u))}{\bigvee_{i=1}^{n} (\varphi(t_i[t/x, u/z]) \wedge \varphi_i[t/x, u/z])} \Updownarrow$$

where $\{d(f(x), z) \equiv t_1 \Leftarrow \varphi_1, \ \dots, \ d(f(x), z) \equiv t_n \Leftarrow \varphi_n\}$
is the coinductive axiomatization of f

optimized unfolding of a μ-predicate r

$$\frac{r(t)}{\bigvee_{i=1}^{k} \exists Z_i : (t \equiv t_i \wedge \varphi_i)} \Updownarrow$$

if $r(t_1) \Leftarrow \varphi_1, \ldots, r(t_n) \Leftarrow \varphi_n$ are the axioms for r,
$Z_i = var(r(t_i) \Leftarrow \varphi_i)$ and $t \equiv t_{k+1}, \ldots, t \equiv t_n$ are unsolvable

optimized unfolding of a ν-predicate r

$$\frac{r(t)}{\bigwedge_{i=1}^{k} \forall Z_i : (t \equiv t_i \Rightarrow \varphi_i)} \Updownarrow$$

if $r(t_1) \Rightarrow \varphi_1, \ldots, r(t_n) \Rightarrow \varphi_n$ are the axioms for r,
$Z_i = var(r(t_i) \Rightarrow \varphi_i)$ and $t \equiv t_{k+1}, \ldots, t \equiv t_n$ are unsolvable

optimized unfolding of a defined function f

$$\frac{\varphi(f(t))}{\bigvee_{i=1}^{n} \exists Z_i : (t \equiv t_i \wedge \psi_i \wedge \varphi(u_i))} \Updownarrow \quad \textit{if SP is functional}$$

if $f(t_1) \equiv u_1 \Leftarrow \psi_1, \ldots, f(t_n) \equiv u_n \Leftarrow \psi_n$ are the axioms for f,
$Z_i = var(f(t_i) \equiv u_i \Leftarrow \psi_i)$ and $t \equiv t_{k+1}, \ldots, t \equiv t_n$ are unsolvable

elimination of a constructor f

$$\frac{f(t_1,\ldots,t_n) \equiv f(u_1,\ldots,u_n)}{t_1 \equiv u_1 \wedge \cdots \wedge t_n \equiv u_n} \Updownarrow$$

if SP is functional

$$\frac{f(t_1,\ldots,t_n) \not\equiv f(u_1,\ldots,u_n)}{t_1 \not\equiv u_1 \vee \cdots \vee t_n \not\equiv u_n} \Updownarrow$$

clash of two different constructors f, g

$$\frac{f(t) \equiv g(u)}{FALSE} \Updownarrow \qquad \frac{f(t) \not\equiv g(u)}{TRUE} \Updownarrow$$

if SP is functional

elimination of an object constructor f

$$\frac{f(t_1,\ldots,t_n) \sim f(u_1,\ldots,u_n)}{t_1 \sim u_1 \wedge \cdots \wedge t_n \sim u_n} \Updownarrow$$

if SP is functional and behaviorally consistent

$$\frac{f(t_1,\ldots,t_n) \not\sim f(u_1,\ldots,u_n)}{t_1 \not\sim u_1 \vee \cdots \vee t_n \not\sim u_n} \Updownarrow$$

clash of two different object constructors f, g

$$\frac{c(t) \sim d(u)}{FALSE} \Updownarrow \qquad \frac{c(t) \not\sim d(u)}{TRUE} \Updownarrow$$

if SP is functional

term replacement

$$\frac{t \equiv u \wedge \varphi(t)}{t \equiv u \wedge \varphi(u)} \Updownarrow \qquad\qquad \frac{t \sim u \wedge \varphi(t)}{t \sim u \wedge \varphi(u)} \Updownarrow \quad \textit{if } \varphi \textit{ is poly-modal}$$

equation elimination

$$\frac{t \equiv u \wedge \varphi(t)}{\varphi(u)} \Downarrow \qquad\qquad \frac{(t \equiv u \wedge \varphi(t)) \;\Rightarrow\; \psi(t)}{\varphi(u) \;\Rightarrow\; \psi(u)} \Uparrow$$

$$\frac{t \sim u \wedge \varphi(t)}{\varphi(u)} \Downarrow \quad \textit{if } \varphi \textit{ is poly-modal}$$

$$\frac{(t \sim u \wedge \varphi(t)) \;\Rightarrow\; \psi(t)}{\varphi(u) \;\Rightarrow\; \psi(u)} \Uparrow \quad \textit{if } \varphi \textit{ and } \psi \textit{ are poly-modal}$$

variable elimination

$$\frac{\exists x : (x \equiv t \wedge \varphi(x))}{\varphi(t)} \Updownarrow \quad \textit{if } x \in X \setminus var(t)$$

$$\frac{\forall x : ((x \equiv t \wedge \varphi(x)) \Rightarrow \psi(x))}{\varphi(t) \Rightarrow \psi(t)} \Updownarrow \quad \textit{if } x \in X \setminus var(t)$$

quantor elimination

$$\frac{\exists x : \varphi(x)}{\varphi(t)} \Uparrow \qquad\qquad \frac{\forall x : \varphi(x)}{\varphi(t)} \Downarrow$$

Boolean rules

$$\frac{\varphi_1 \Rightarrow (\varphi_2 \Rightarrow \psi)}{(\varphi_1 \wedge \varphi_2) \Rightarrow \psi} \Updownarrow$$

$$\frac{\varphi \Rightarrow (\psi_1 \wedge \psi_2)}{(\varphi \Rightarrow \psi_1) \wedge (\varphi \Rightarrow \psi_2)} \Updownarrow$$

$$\frac{(\varphi_1 \vee \varphi_2) \Rightarrow \psi}{(\varphi_1 \Rightarrow \psi) \wedge (\varphi_2 \Rightarrow \psi)} \Updownarrow$$

$$\frac{\bigwedge_{i=1}^{n}(\varphi_i \Rightarrow \psi_i) \quad \Rightarrow \quad \psi}{\bigvee_{i=1}^{n}(\varphi_i \wedge (\psi_i \Rightarrow \psi)) \vee \psi} \Updownarrow$$

$$\frac{\varphi \Rightarrow \bigvee_{i=1}^{n}(\varphi_i \Rightarrow \psi_i)}{(\varphi \wedge \bigwedge_{i=1}^{n} \varphi_i) \Rightarrow \bigvee_{i=1}^{n} \psi_i} \Updownarrow$$

subsumption

$$\frac{\varphi_1 \vee \cdots \vee \varphi_m \quad \Rightarrow \quad \exists X_1 \psi_1 \vee \cdots \vee \exists X_n \psi_n}{\mathit{TRUE}} \Updownarrow$$

if the premise subsumes the conclusion, i.e.

$$\left.\begin{array}{l} \forall\, 1 \leq i \leq m \\ \exists\, 1 \leq k \leq n,\ \sigma : X_k \to T_\Sigma(X),\ \psi \end{array}\right\} \quad \varphi_i = (\psi_k \sigma \wedge \psi).$$

fixpoint induction on a μ-predicate r

$$\frac{r(x) \Rightarrow \psi}{\exists q : q(x) \Rightarrow \psi \;\wedge\; \bigwedge_{\varphi \in AX_r} \varphi[q/r]} \Updownarrow \qquad \begin{array}{l} q \; \textit{strengthens} \; \psi : \\ r(x) \Rightarrow q(x) \Rightarrow \psi \end{array}$$

simultaneous fixpoint induction on $r_1, \ldots, r_n$

$$\frac{\bigwedge_{i=1}^{n}(r_i(x) \Rightarrow \psi_i)}{\exists q_1, \ldots, q_n : \bigwedge_{i=1}^{n}(q_i(x) \Rightarrow \psi_i \;\wedge\; \bigwedge_{\varphi \in AX_{r_i}} \varphi[q_j/r_j \mid 1 \leq j \leq n])} \Updownarrow$$

fixpoint induction on $\Diamond r$

$$\frac{(\Diamond r)(t) \Rightarrow \psi}{\exists q : \; q(t) \Rightarrow \psi \;\; \wedge \;\; r(x) \Rightarrow q(x) \;\; \wedge \;\; (x \to y \wedge q(y)) \Rightarrow q(x)} \Updownarrow$$

fixpoint induction on a defined function $f : w \to s$

$$\frac{f(x) \equiv y \Rightarrow \psi}{\exists q : ws : q(x,y) \Rightarrow \psi \;\wedge\; \bigwedge_{\varphi \in flat(AX_f)} \varphi[q/\lambda(x,y).f(x) \equiv y]} \Updownarrow$$

if SP is functional

coinduction on a ν-predicate r

$$\frac{\psi \Rightarrow r(x)}{\exists q : \psi \Rightarrow q(x) \ \wedge \ \bigwedge_{\varphi \in AX_r} \varphi[q/r]} \Updownarrow$$

q *weakens* ψ :

$\psi \Rightarrow q(x) \Rightarrow r(x)$

simultaneous coinduction on $r_1, \dots, r_n$

$$\frac{\bigwedge_{i=1}^{n}(\psi_i \Rightarrow r_i(x))}{\exists q_1, \dots, q_n : \bigwedge_{i=1}^{n}(\psi_i \Rightarrow q_i(x) \ \wedge \ \bigwedge_{\varphi \in AX_{r_i}} \varphi[q_j/r_j \mid 1 \le j \le n])} \Updownarrow$$

coinduction on $\Box r$

$$\frac{\psi \Rightarrow (\Box r)(t)}{\exists q : \ \psi \Rightarrow q(t) \ \ \wedge \ \ q(x) \Rightarrow r(x) \ \ \wedge \ \ (x \to y \wedge q(x)) \Rightarrow q(y)} \Updownarrow$$

coinduction on $\sim$

$$\frac{\bigwedge_{s \in S} \psi \Rightarrow t_s \sim_s u_s}{\exists \{q_s : ss\}_{s \in S} : \bigwedge_{s \in S} \begin{cases} \psi \Rightarrow eq(q_s)(t_s, u_s) \quad \wedge \\ \bigwedge_{(x \sim_s y \Rightarrow \varphi) \in AX}(q_s(x,y) \Rightarrow \varphi[eq(q_s)/\sim_s \mid s \in S]) \end{cases}} \Updownarrow$$

where for all $s \in S$, $eq(q_s)$ is the equivalence closure of q_s

Let δ be of the form

$$\exists Y\varphi \quad \Rightarrow \quad \bigvee_{i=1}^{k} \exists X_i u_i \vee \bigvee_{i=k+1}^{m} \exists X_i u \equiv u_i \vee \bigvee_{i=m+1}^{n} \exists X_i u' \sim u_i.$$

contraction with δ = forward application of δ

$$\frac{\exists Y(\varphi\sigma \wedge \psi(u\sigma) \wedge \psi(u'\sigma))}{\bigvee_{i=1}^{k} \exists X_i(\psi \wedge u_i\sigma) \vee \bigvee_{i=k+1}^{m} X_i\psi(u_i\sigma) \vee \bigvee_{i=m+1}^{n} X_i\psi(u_i\sigma)} \Downarrow$$

if for all $1 \leq i \leq n$, $(X_i \cup Y) \cap dom(\sigma) = \emptyset$
(and the premise is poly-modal)

expansion with δ = backward application of δ

$$\frac{\bigvee_{i=1}^{k} \exists X_i(\psi_i \wedge t_i) \ \vee \ \bigvee_{i=k+1}^{m} \exists X_i\psi_i(t_i) \ \vee \ \bigvee_{i=m+1}^{n} \exists X_i\psi_i(t_i)}{\exists \biguplus_{i=1}^{n} X_i \uplus Y \left(\begin{array}{l} \varphi\sigma \wedge \bigwedge_{i=1}^{k} \psi_i\sigma \wedge \bigwedge_{i=k+1}^{n} \psi_i(u)\sigma \wedge \\ \bigwedge_{i=m+1}^{n} \psi_i(u')\sigma \wedge \bigwedge_{x\in V} x \equiv x\sigma \end{array} \right)} \Uparrow$$

where V is the set of free variables of the premise
and for all $1 \leq i \leq n$,
$t_i\sigma = u_i\sigma$ and $(X_i \cup Z) \cap dom(\sigma) = \emptyset = X_i \cap var(t_i) \cap var(\psi_i)$
(and the premise is poly-modal)

Let δ be of the form

$$\exists Y\varphi \quad \Rightarrow \quad \bigvee_{i=1}^{k} \exists X_i u_i \vee \bigvee_{i=k+1}^{m} \exists X_i u \equiv u_i \vee \bigvee_{i=m+1}^{n} \exists X_i u' \sim u_i,$$

$\vec{x}$ be a list of all free variables of F, $\gg: sort(\vec{x})^2 \in \Sigma$ and $\rho : X \to X$ be a renaming of variables such that $\gg^{Ini(SP)}$ is well-founded.

inductive contraction with δ

$$\frac{\exists Y(\varphi\sigma \wedge \psi(u\sigma) \wedge \psi(u'\sigma)) \wedge \vec{x} \gg \vec{x}\rho\sigma}{\bigvee_{i=1}^{k} \exists X_i(\psi \wedge u_i\sigma) \vee \bigvee_{i=k+1}^{m} X_i\psi(u_i\sigma) \vee \bigvee_{i=m+1}^{n} X_i\psi(u_i\sigma)} \Downarrow$$

if for all $1 \leq i \leq n$, $(X_i \cup Y) \cap dom(\sigma) = \emptyset$
(and the premise is poly-modal)

inductive expansion with δ

$$\frac{\bigvee_{i=1}^{k} \exists X_i(\psi_i \wedge t_i) \ \vee \ \bigvee_{i=k+1}^{m} \exists X_i\psi_i(t_i) \ \vee \ \bigvee_{i=m+1}^{n} \exists X_i\psi_i(t_i)}{\exists \biguplus_{i=1}^{n} X_i \uplus Y(\begin{array}{l} \varphi\sigma \wedge \bigwedge_{i=1}^{k} \psi_i\sigma \wedge \bigwedge_{i=k+1}^{n} \psi_i(u)\sigma \wedge \\ \bigwedge_{i=m+1}^{n} \psi_i(u')\sigma \wedge \bigwedge_{x\in V} x \equiv x\sigma \wedge \vec{x} \gg \vec{x}\rho\sigma \end{array})} \Uparrow$$

where V is the set of free variables of the premise
and for all $1 \leq i \leq n$,
$t_i\sigma = u_i\sigma$ and $(X_i \cup Z) \cap dom(\sigma) = \emptyset = X_i \cap var(t_i) \cap var(\psi_i)$
(and the premise is poly-modal)

Loop invariants

An *iterative program* for a function $f : I \to O$ is a set of axioms of the following form:

$$
\begin{aligned}
&f(x) \equiv g(in(x)) \\
&g(y) \equiv g(loop_1(y)) \;\Leftarrow\; \varphi_1(y) \\
&\dots \\
&g(y) \equiv g(loop_n(y)) \;\Leftarrow\; \varphi_n(y) \\
&g(y) \equiv out(y) \;\Leftarrow\; \varphi_{n+1}(y)
\end{aligned}
$$

$in : I \to L$, $loop_1, \dots, loop_n : L \to L$ and $g, out : L \to O$ are auxiliary functions using the *loop domain* L.

The existence of

- a *Hoare invariant* $hoi : I \times L$ or
- a *subgoal invariant* $sub : L \times O$.

guarantees that the graph of $f : I \to O$ equal to a given input-output relation $r : I \times O$.

sub is a subgoal invariant if sub satisfies the conclusion of subgoal induction, which is simultaneous fixpoint induction on the graphs of f and g:

subgoal induction

$$\frac{\begin{array}{rcl} r(x,z) & \Leftarrow & f(x) \equiv z \\ \wedge\ sub(y,z) & \Leftarrow & g(y) \equiv z \end{array}}{\begin{array}{lrcl} & r(x,z) & \Leftarrow & sub(in(x),z) \\ \wedge & sub(y,z) & \Leftarrow & sub(loop_1(y),z) \wedge \varphi_1(y) \\ \ldots & & & \\ \wedge & sub(y,z) & \Leftarrow & sub(loop_n(y),z) \wedge \varphi_n(y) \\ \wedge & sub(y,out(y)) & \Leftarrow & \varphi_{n+1}(y) \end{array}} \Uparrow$$

hoi is a Hoare invariant if hoi satisfies the conclusion of Hoare induction:

Hoare induction

$$\frac{r(x,z) \ \Leftarrow\ f(x) \equiv z}{\begin{array}{lrcl} & hoi(x,in(x)) & & \\ \wedge & hoi(x,loop_1(y)) & \Leftarrow & hoi(x,y) \wedge \varphi_1(y) \\ \ldots & & & \\ \wedge & hoi(x,loop_n(y)) & \Leftarrow & hoi(x,y) \wedge \varphi_n(y) \\ \wedge & r(x,out(y)) & \Leftarrow & hoi(x,y) \wedge \varphi_{n+1}(y) \end{array}} \Uparrow$$

A Hoare invariant defines a subgoal invariant:

$$sub(y,z) \iff \forall x : r(x,z) \Leftarrow hoi(x,y).$$

A subgoal invariant defines a Hoare invariant:

$$hoi(x,y) \iff \forall z : r(x,z) \Leftarrow sub(y,z).$$

Basic category theory

relational theory	category theory	
set inclusion	morphism	
$A \subseteq B$	$A \to B$	
union	colimes	
intersection	limes	
least relation	initial object	
greatest relation	final object	
monotone function F	functor F	monad M
$A \subseteq B$	$A \xrightarrow{f} B$	$A \xrightarrow{f} B \xrightarrow{unit_B} M(B)$
$F(A) \subseteq F(B)$	$F(A) \xrightarrow{F(f)} F(B)$	$M(A) \xrightarrow{(unit_B \circ f)^*} M(B)$
upward continuous function	ω-colimes-preserving $=$ cocontinuous functor	
downward continuous function	ω-limes-preserving $=$ continuous functor	

Algebras and coalgebras

Let $\mathcal{K}$ be a category and $F:\mathcal{K} \to \mathcal{K}$ be a functor.

An F-*algebra* or F-*dynamics* is a $\mathcal{K}$-morphism

$$\alpha : F(A) \to A.$$

An F-*coalgebra* or F-*codynamics* is a $\mathcal{K}$-morphism

$$\beta : B \to F(B).$$

A *fixpoint of* F is a $\mathcal{K}$-object A with $F(A) \cong A$.

If some $F(A) \to A$ is initial in $Alg(F)$, then A is a fixpoint of F, called the *least fixpoint* of F.
If some $A \to F(A)$ is final in $coAlg(F)$, then A is a fixpoint of F, called the *greatest fixpoint* of F.

F is *upward continuous* or *cocontinuous* if F preserves
colimits of ascending chains in $\mathcal{K}$.
F is *downward continuous* if F preserves
limits of descending chains in $\mathcal{K}$.

F is *bicontinuous* if F is up- and downward continuous.

Categorical version of Kleene's fixpoint theorem

Suppose that $\mathcal{K}$ has an initial object I and colimits of ascending chains. Let $F : \mathcal{K} \to \mathcal{K}$ be an upward continuous functor and A be the colimit of the chain

$$I \to F(I) \to F^2(I) \longrightarrow \ldots .$$

Then $F(A)$ is the colimit of

$$F(I) \longrightarrow F^2(I) \to F^3(I) \to \ldots$$

and the unique $\mathcal{K}$-morphism $F(A) \to A$ is initial in $Alg(F)$.

Suppose that $\mathcal{K}$ has a final object T and limits of descending chains. Let $F : \mathcal{K} \to \mathcal{K}$ be a downward continuous functor and B be the limit of the chain

$$T \leftarrow F(T) \leftarrow F^2(T) \leftarrow \ldots .$$

Then $F(B)$ is the limit of

$$F(T) \leftarrow F^2(T) \leftarrow F^3(T) \leftarrow \ldots$$

and the unique $\mathcal{K}$-morphism $B \leftarrow F(B)$ is final in $coAlg(F)$.

All endofunctors on Set^S built up from the identity functor, constant functors, coproducts and finite (or infinite) products are upward (and downward) continuous.

Bicontinuous term functors F

Let con_Σ be a signature of constructors with hidden range sort and C be a $visS$-sorted set.

$F : Set^S \to Set^S$ is defined as follows: for all $A \in Set^S$ and $s \in S$,

$$F(A)_s = \begin{cases} C_s & \text{if } s \in visS, \\ \coprod_{f w \to s \in con_\Sigma} A_w & \text{if } s \in hidS. \end{cases}$$

F is upward continuous.

☞ $Alg(F)$ has an initial object $ini : F(Ini(F)) \to Ini(F)$.

$Ini(F)$ is the algebra $T_{con\Sigma \cup C}$ of finite ground terms over $con_\Sigma \cup C$.

Examples:

$$\begin{aligned} Nat(A)_{nat} &= 1 + A_{nat}, \\ Ini(Nat)_{nat} &= \{succ^n(0) \mid n \in \mathbb{N}\}, \end{aligned}$$

$$\begin{aligned} List(A)_{list(s)} &= 1 + (C_s \times A_{list(s)}), \\ Ini(List)_{list(s)} &= \{a_1 :: \cdots :: a_n :: nil \mid a_1, \ldots, a_n \in C_s,\ n \in \mathbb{N}\}. \end{aligned}$$

F is downward continuous.

☞ $coAlg(F)$ has a final object $fin : Fin(F) \to F(Fin(F))$.

$Fin(F)$ is the algebra $T^\infty_{con\Sigma\cup C}$ of finite or infinite ground terms over $con_\Sigma \cup C$.

$T^\infty_{con\Sigma\cup C}$ is a subset of $T_{con\Sigma\cup C} \cup [\mathbb{N}^* \to (con_\Sigma \cup C)]$.

Examples:

$$\begin{aligned} Fin(Nat)_{nat} &= Ini(Nat)_{nat} \cup \{succ^\infty(0)\} \\ &= \mathbb{N} \cup \{\infty\}, \\ Fin(List)_{list(s)} &= Ini(List)_{list(s)} \\ &\quad \cup \{a_1 :: a_2 :: \dots \mid a_1, a_2, \dots \in C_s\}. \end{aligned}$$

Downward continuous coterm functors G

Let des_Σ be a signature of *destructors* with a unique hidden argument and C be a $visS$-sorted set.

$G : Set^S \to Set^S$ is defined as follows: for all $A \in Set^S$ and $s \in S$,

$$G(A)_s \;=\; \begin{cases} C_s & \text{if } s \in visS, \\ \prod_{fsw \to s' \in des_\Sigma} [C_w \to A_{s'}] & \text{if } s \in hidS. \end{cases}$$

Derived destructors:

- $des_\Sigma \;\subseteq\; des^+_\Sigma$.
- For all $c : w \to s$ and $d : sw' \to s' \in des^+_\Sigma$,
  $d \cdot c : ww' \to s' \in des^+_\Sigma$.

Σ-context: derived Σ-destructor $d : w \to s$ with $s \in visS$.

G is downward continuous.
☞ $coAlg(G)$ has a final object $fin : Fin(G) \to G(Fin(G))$.

$Fin(G)$ is a product of function spaces: for all $s \in S$,

$$Fin(G)_s \;=\; \begin{cases} C_s & \text{if } s \in visS, \\ \prod_{c:sw \to s' \;\; \Sigma-context} [C_w \to C_{s'}] & \text{if } s \in hidS. \end{cases}$$

Examples:

$$\begin{aligned}
G(A)_{set(s)} &= [C_s \to 2], \\
G(A)_{bag(s)} &= [C_s \to \mathbb{N}], \\
G(A)_{weighted_set(s)} &= [C_s \to \mathbb{Z}], \\
G(A)_{map(dom,ran)} &= [C_{dom} \to C_{ran}], \\
G(A)_{mealy(s,state,s')} &= [C_s \to A_{state}] \times [C_s \to C_{s'}].
\end{aligned}$$

If the range of a destructor is a sum,
then $Fin(G)$ *is a subset of* $\prod_{c:sw\to s' \;\Sigma-context}[C_w \to C_{s'}]$.

Examples:

$$\begin{aligned}
G(A)_{partial_map(dom,ran)} &= [C_{dom} \to 1 + C_{ran}], \\
G(A)_{partial_mealy(s,state,s')} &= [C_s \to 1 + A_{state}] \times [C_s \to 1 + C_{s'}], \\
G(A)_{stream(s)} &= 1 + (C_s \times A_{stream(s)}), \\
G(A)_{bintree(s)} &= C_s \times (1 + (A_{bintree(s)} \times A_{bintree(s)})).
\end{aligned}$$

Cosignature $\Delta = (vis\Sigma, hidS, des)$

$vis\Sigma = (visS, F, P)$ *swinging signature*

$visS$ *visible sorts*

$hidS$ *hidden sorts*

des *destructors* $d : s \to (w \to \amalg_{i \in I} w_i)$

where $s \in hidS$, $w \in visS^*$, $w_i \in S^+$

$ID = \{id : s \to s \mid s \in visS\}$

The set cT_Δ of Δ-*coterms*:

- $des \subseteq cT_\Delta$.
- For all I and $\{e_i : s_i \to s_i'\}_{i \in I} \subseteq cT_\Delta \cup ID$,
$$(\amalg_{i \in I} e_i) : \amalg_{i \in I} s_i \to \amalg_{i \in I} s_i' \in cT_\Delta.$$
- For all $s' \in hidS$, $d : s \to (w \to s') \in des$ and $e : s' \to s'' \in cT_\Delta$,
$$e \cdot d : s \to (w \to s'') \in cT_\Delta.$$

The set CT_Δ of Δ-*contexts* consists of all Δ-coterms $e : s \to (w \to s')$ such that $s' \in visS$.

(Δ, C)-coalgebra

Let C be a $vis\Sigma$-structure with equality, i.e.

- either $vis\Sigma$ is basic Horn and $a \equiv^C b$ iff $a = b$
- or $a \sim^C b$ iff $a = b$.

A *(Δ, C)-coalgebra A* is a $(visS \cup hidS, F \cup des, P)$-structure such that $A|_{vis\Sigma} = C$.

For all sorts s, let

$$P_s \;=_{def}\; \begin{cases} C_s & \text{if } s \in visS, \\ \prod_{d:s\to(w\to s')\in CT_\Delta}[C_w \to C_{s'}] & \text{if } s \in hidS. \end{cases}$$

Final (Δ, C)**-coalgebra** $Fin(\Delta, C)$

$\Phi : \wp(P) \to \wp(P)$ is defined as follows: for all $s \in S$ and $A \subseteq P$,

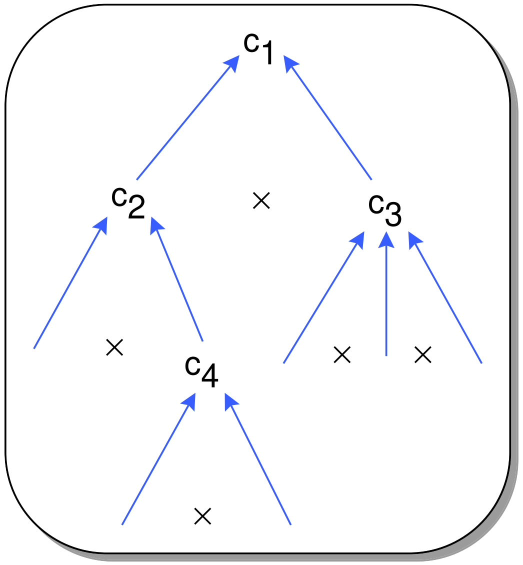
$$\Phi(A)_s = \begin{cases} C_s & \text{if } s \in visS, \\ \{a \in A_s \mid \left\{ \begin{array}{l} \forall\, s' \in hidS, \\ \forall\, d : s \to (w \to s') \\ \qquad \in des, \\ \forall\, c \in C_w\ \exists\, b_a \in A_{s'} \\ \forall\, e : s' \to s'' \in CT_\Delta : \\ \pi_{e \cdot d}(a)(c) = \pi_e(b_a) \end{array} \right\} \} & \text{if } s \in hidS. \end{cases}$$

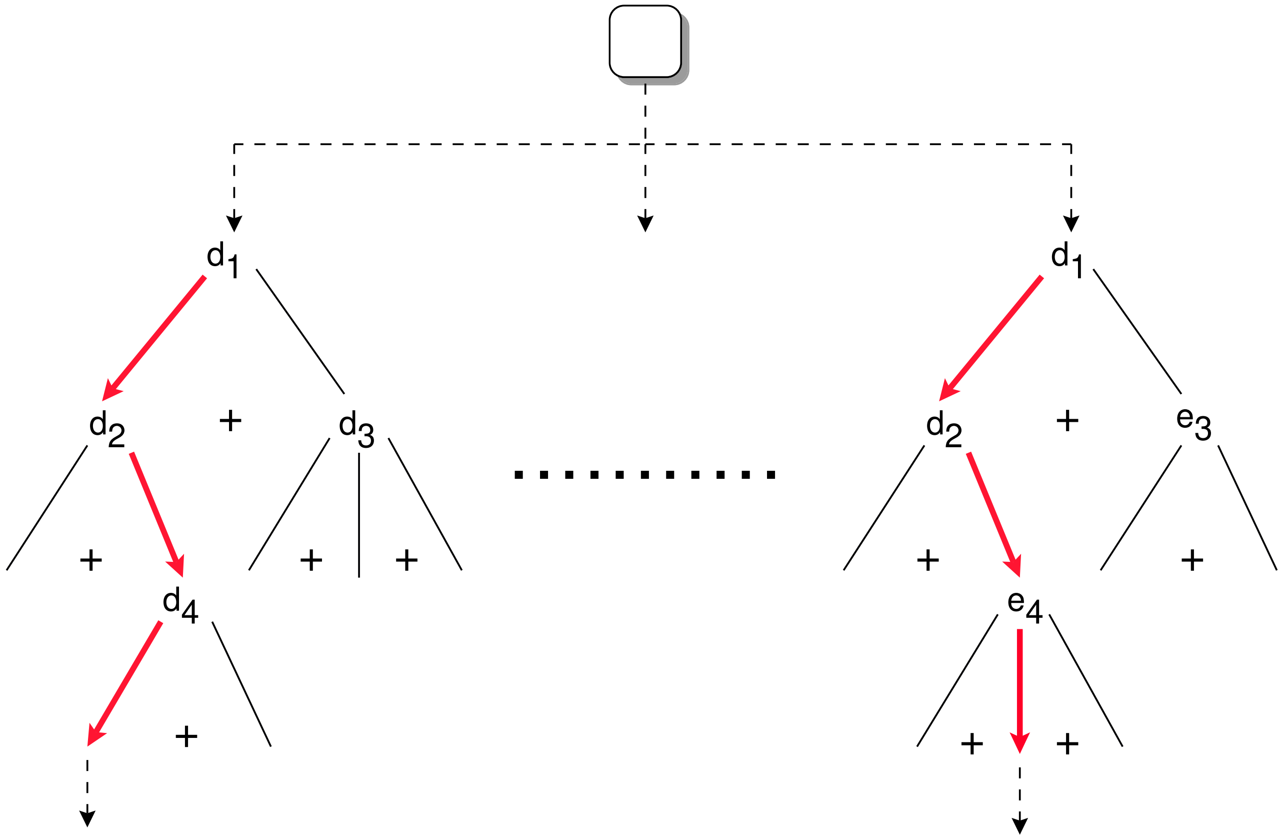
$Fin(\Delta, C)$ is the (Δ, C)-coalgebra such that

- for all $s \in hidS$, $Fin_s = gfp(\Phi)_s$,
- for all $d : s \to (w \to s') \in des$, $a \in Fin_s$ and $c \in C_w$,

$$d^{Fin}(a)(c) = \begin{cases} \pi_d(a)(c) & \text{if } s' \in visS, \\ b_a & \text{if } s' \in hidS. \end{cases}$$

If all Δ*-contexts have coarity 1, then* Fin_s *agrees with* P_s*.*





term

consisting of constructors

behavior

consisting of context interpretations

Cospecification

$$CSP \;=\; (\Delta, C, aux, CP, cof, AX_{aux \cup CP \cup cof})$$

$\Delta = (vis\Sigma, hidS, des)$	*cosignature*
C	*$vis\Sigma$-structure with equality*
$aux \ni f : sw \to s'$	*auxiliary function* $\;\Rightarrow\; s \in visS$
AX_{aux}	*inductive axiomatization of aux*
$\{\sim_s : ss \mid s \in hidS\}$	*behavioral equalities*
νP	*ν-predicates*
AX_{hidS}	*assertions*
$AX_{\nu P}$	*co-Horn clauses*
$cof \ni f : w \to s$	*cofunction* $\;\Rightarrow\; s \in hidS$
AX_{cof}	*coinductive axiomatization of cof*

inductive axiomatization of $f : sw \to s'$:

$$
\begin{aligned}
AX_f &= \cup_{t \in PAT} AX_{f,t} \\
AX_{f,t} &= \{f(t,x) \equiv t_1 \;\Leftarrow\; \varphi_1, \\
&\qquad \ldots, \\
&\qquad f(t,x) \equiv t_n \;\Leftarrow\; \varphi_n\}
\end{aligned}
$$

- $t_1, \ldots, t_n$ and $\varphi_1, \ldots, \varphi_n$ do not contain ν-predicates or cofunctions,
- for all $1 \leq i \leq n$, $g \in aux$, subterms $g(u,v)$ of t_i or φ_i and $b : X \to C$, $b^*(t) > b^*(u)$,
- $coAlg(\Delta, C)$ satisfies $\varphi_1 \overset{\bullet}{\vee} \ldots \overset{\bullet}{\vee} \varphi_n$.

axiomatization of behavioral equalities:

$$x \sim y \Rightarrow d(x)(z) \sim d(y)(z)$$
for all $d\colon s \to (w \to s') \in des$ and $s' \in hidS$

assertions:

$$G \ \Rightarrow \ \varphi$$

- $G \Rightarrow \varphi$ does not contain cofunctions,
- G does not contain ν-predicates.

(Implicit) invariance axioms:

$$is_s(x) \Rightarrow (d(x)(z) \equiv \kappa_i(y) \Rightarrow is_{w_i}(y))$$
for all $d\colon s \to (w \to \coprod_{i \in I} w_i) \in des$ and $w_i \in hidS$

$$is_{s_1 \times \cdots \times s_n}(x_1, \ldots, x_n) \Rightarrow is_{s_i}(x_i)$$
for all $s_1, \ldots, s_n \in S$ and $s_i \in hidS$

coinductive axiomatization of $f : w \to s$:

$$
\begin{aligned}
AX_f &= \cup_{d:s\to s'\in des} AX_{f,d} \\
AX_{f,d} &= \{d(f(x))(z) \equiv t_1 \;\Leftarrow\; \varphi_1, \\
&\quad\;\; \ldots, \\
&\quad\;\; d(f(x))(z) \equiv t_n \;\Leftarrow\; \varphi_n\}
\end{aligned}
$$

- $\varphi_1, \ldots, \varphi_n$ do not contain ν-predicates or cofunctions,
- $Fin(\Delta, C)$ satisfies $\varphi_1 \stackrel{\bullet}{\vee} \ldots \stackrel{\bullet}{\vee} \varphi_n$.

CSP-model

Let $\Sigma = vis\Sigma \cup (hidS, des \cup aux, CP)$.

A Σ-structure A with $\equiv$-equality is a *CSP-model* if A satisfies $AX_{aux \cup CP}$ and the congruence axioms for Σ.

A CSP-model A is a *CSP-model with is-membership* if for all $s \in hidS$, $is_s^A = A_s$.

Each (Δ, C)-coalgebra A can be extended to a CSP-model:

- aux is interpreted by the unique solution of AX_{aux} in A,
- CP is interpreted by the greatest solution of AX_{CP} in A.

In particular, $Fin(\Delta, C)$ is a CSP-model.

$Fin(\Delta, C)$ interprets behavioral equalities as identities.

Final CSP**-model** $Fin(CSP)$

$Fin(CSP)$ is the subcoalgebra of $Fin(\Delta, C)$ such that for all $s \in hidS$,

$$Fin(CSP)_s \;=\; is_s^{Fin(\Delta,C)}.$$

φ is a *coinductive theorem of* CSP if $Fin(CSP) \models \varphi$.

$Fin(CSP)$ is final in the class of CSP-models with is-membership.

Algebraic specifications are function-oriented.

A signature Σ defines functions on objects that are terms consisting of constructors.

☞ *The standard model of Σ*
is an initial algebra of terms,
i.e. a sum of products.

The axioms of a specification SP with signature Σ identify objects and specify functions inductively.

☞ *The standard model of SP*
is a quotient *of the standard model of Σ.*

Coalgebraic specifications are object-oriented.

A cosignature Δ defines object states as contexts consisting of destructors.

☞ *The standard model of Δ is a final coalgebra of object states, i.e. a product of function spaces.*

The axioms of a cospecification CSP with cosignature Δ select objects and specify functions coinductively.

☞ *The standard model of CSP is a* subcoalgebra *of the standard model of Δ.*

Σ	signature	Δ	cosignature
$I = Ini(\Sigma)$	initial Σ-algebra	$F = Fin(\Delta, C)$	final (Δ, C)-coalgebra
AX	Horn clauses	AX	co-Horn clauses
$SP = (\Sigma, AX \cup \dots)$	swinging specification	$CSP = (\Delta, C, \dots, AX)$	cospecification
$Mod(SP)_\equiv$ $= \{A \in Mod(SP) \mid \forall\ s \in S : \equiv^A = \{(a,a) \mid a \in A_s\}\}$		$Mod(CSP)_{is}$ $= \{A \in Mod(CSP) \mid \forall\ s \in hidS : is_s^A = A_s\}$	
$I \xrightarrow{ini} A$ $kernel(ini) \models AX \cup EQ_\Sigma$ $\equiv^I =_{def}$ least subset of $I \times I$ s.t. $I \models AX \cup EQ_\Sigma$		$A \xrightarrow{fin} F$ $image(fin) \models AX \cup MP_\Delta$ $is^F =_{def}$ greatest subset of F s.t. $F \models AX \cup MP_\Delta$	
$\Longrightarrow \quad \equiv^I \subseteq kernel(ini)$		$\Longrightarrow \quad image(fin) \subseteq is^F$	
$\Longleftrightarrow \quad ini$ is compatible with $\equiv$		$\Longleftrightarrow \quad fin$ is compatible with is	
$\Longrightarrow \begin{cases} I \xrightarrow{nat} I/\equiv^I \xrightarrow{ini'} A \quad ini' \text{ is} \\ \qquad [t] \mapsto ini(t) \quad \text{well-defined} \end{cases}$		$\Longrightarrow \begin{cases} A \xrightarrow{fin'} is^F \xrightarrow{inc} F \quad fin' \text{ is} \\ a \mapsto fin(a) \qquad \text{well-defined} \end{cases}$	
$\overset{I \in Mod(SP)}{\Longrightarrow} \quad I/\equiv^I$ is initial in $Mod(SP)_\equiv$		$\overset{F \in Mod(CSP)}{\Longrightarrow} \quad is^F$ is final in $Mod(CSP)_{is}$	

Coalgebraic swinging type

$$CST \;=\; (SP, aux, CP, cof, CAX)$$

$SP = (\Sigma, AX)$	*non-basic Horn swinging type*
$hidS$	*hidden sorts of* Σ
$visSP = (vis\Sigma, visAX)$	*visible subspecification of* SP
$\sim_{SP}^{vis}$	*least* Σ*-congruence on* T_Σ
	including $\equiv_{SP}$ *and* $\sim_{visSP}$
$\Delta = (vis\Sigma, hidS, des)$	*cosignature*
$CSP = (\Delta, T_\Sigma/\sim_{SP}^{vis}, aux,$	
$CP, cof, CAX)$	*cospecification*

- des is the set of $hidS$-observers,
- there are no $hidS$-object constructors,
- the auxiliary functions of CSP are hidden functions of Σ,
- the ν-predicates of CSP are ν-predicates of Σ,
- CAX includes all axioms of SP that are axioms for aux or CP.

Domain completion and iceberg model of CST

Suppose that for all $hidS$-destructors $f:sw\rightarrow s'$, s' is hidden or *strongly visible*, i.e. $\equiv_{SP,s}$ agrees with $\sim_{SP,s}$.

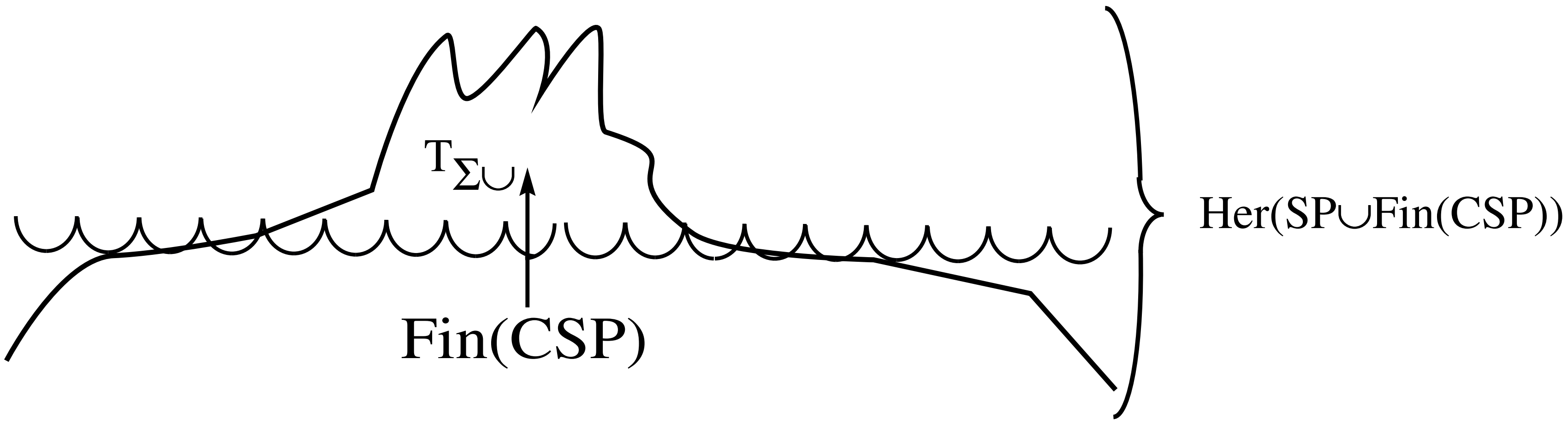
Let $\Sigma'=\Sigma\cup Fin(CSP)$,

$$\begin{array}{rll} AX_{des} & = & \{f(a,u)\equiv t\mid\ f:sw\rightarrow s'\in des,\ s'\in visS, \\ & & \quad a\in Fin(CSP)_s,\ u,t\in NF_\Sigma, \\ & & \quad f^{Fin(CSP)}(a)([u])=[t]\}\ \cup \\ & & \{f(a,u)\equiv b\mid\ f:sw\rightarrow s'\in des,\ s'\in hidS, \\ & & \quad a\in Fin(CSP)_s,\ u\in NF_\Sigma, \\ & & \quad f^{Fin(CSP)}(a)([u])=b\}. \end{array}$$

$SP'=(\Sigma',AX\cup AX_{des})$ *domain completion of* CST

$Her(SP')$ *iceberg model of* CST

SP' is *membership compatible* if for all $s\in hidS$, $is_s^{Her(SP')}=T_{\Sigma',s}$.


T$_{\Sigma\cup}$
Fin(CSP)
Her(SP∪Fin(CSP))

Theorem

Suppose that a coalgebraic swinging type CST is functional, continuous, behaviorally consistent and cospec closed, i.e.

- ➢ *CST has no assertions or*
- ➢ *CST has no hidden constructors or*
- ➢ *all ground hidden normal forms t are SP'-equivalent to some $a \in Fin(CSP)$.*

Let SP' be the domain completion of CST by $Fin(CSP)$.

- *$Fin(SP')$ is isomorphic to $Fin(CSP)$.*
- *For all $a, b \in Fin(CSP)$, $a \sim_{SP'} b$ iff $a = b$.*

$$
\left.\begin{array}{l} SP = (\Sigma, AX) \\ CSP = (\Delta, C, aux, cof, AX') \end{array}\right\} \; CST = (SP, exp, aux, cof, AX')
$$

$$
\begin{tikzcd}
 & & & (NF_{\Sigma'})_{exp'} \;\; \textit{CSP-model} \arrow[dr, "h\;\; \textit{final } (D,C)\textit{-homomorphism}"] & \\
 & & (NF_{\Sigma'})_{exp'} \arrow[ur, equal] \arrow[dr, "h"] & & Fin(CSP) \arrow[dl, "inc"] \\
Ice(CST) = Her(SP')_{exp'} \arrow[urr, "nf"] \arrow[dr, "nat"'] & & & Fin(CST) & \\
 & Ice(CST)/_{kernel(h \circ nf)} = Fin(SP')_{exp'} \arrow[urr, dashed, leftrightarrow] & & &
\end{tikzcd}
$$

$kernel(h \circ nf)$

$= \textit{contextual CST-equivalence} \,|\, T_{exp'}$

$= \textit{behavioral SP}'\textit{-equivalence} \,|\, T_{exp'}$

$SP' = (\Sigma', AX \cup AX_{des_\Sigma})$

$= \textit{blending of CST}$

$\Sigma' = \Sigma \cup Fin(CSP)$

$exp' = exp \cup Fin(CSP)$

The final model of a coalgebraic ST is isomorphic to the final model of its grounding

ENTRY(s)

`preds`	$\_ \not\equiv \_ : s \times s$
`vars`	$x, y : s$
`axioms`	$x \not\equiv y \Leftrightarrow \neg(x \equiv y)$

LIST $=$ **ENTRY**$(entry)$ `then`

`hidsorts`	$list = list(entry)$
`objconstructs`	$nil :\to list$
	$\_ :: \_ : entry \times list \to list$

ACC_STATE $=$ **NAT** `and` **LIST** `then`

`hidsorts`	$transaction \quad acc$
`objconstructs`	$from, to : nat \times entry \to transaction$
	$new : entry \to acc$
	$credit, debit : nat \times acc \times entry \to acc$
`destructs`	$name : acc \to entry$
	$bal : acc \to nat$
	$record : acc \to list(transaction)$
`vars`	$x : entry \quad n : nat \quad a : acc$
`Horn axioms`	$name(new(x)) \equiv x$
	$bal(new(x)) \equiv 0$
	$record(new(x)) \equiv nil$
	$name(credit(n, a, x)) \equiv name(a)$
	$bal(credit(n, a, x)) \equiv bal(a) + n$
	$record(credit(n, a, x)) \equiv from(x, n) :: record(a)$
	$name(debit(n, a, x)) \equiv name(a)$
	$bal(debit(n, a, x)) \equiv bal(a) - n$
	$record(debit(n, a, x)) \equiv to(x, n) :: record(a)$

ACC_LOCAL1 $=$ **ACC_STATE** `then`

`hidsorts`	com
`objconstructs`	$deposit, withdraw : nat \to com$
	$send, receive : nat \times entry \to com$
`defuncts`	$\_ : \_ : acc \times com \to acc$
`vars`	$x : entry \quad n : nat \quad a : acc \quad c : com$
`Horn axioms`	$a : deposit(n) \equiv credit(a, n, name(a))$
	$a : withdraw(n) \equiv debit(a, n, name(a)) \quad \Leftarrow \quad n \leq bal(a)$
	$a : withdraw(n) \equiv a \quad \Leftarrow \quad n > bal(a)$
	$a : send(n, x) \equiv debit(a, n, x) \quad \Leftarrow \quad n \leq bal(a)$
	$a : send(n, x) \equiv debit(a, n, x) \quad \Leftarrow \quad n > bal(a)$
	$a : receive(n, x) \equiv credit(a, n, x)$

ACC_GLOBAL1 = ACC_LOCAL1 `and` **SET** `then`

```
hidsorts         message
objconstructs    open, close : entry → message
                 _._ : entry × com → message
                 _; _ : message × message → message
defuncts         _ : _ : set(acc) × message → set(acc)
                 transfer : entry × nat × entry → message
static preds     _ ∈ _ : entry × set(acc)
                 _ ∉ _ : entry × set(acc)
ν-preds          _ ≈ _ : message × message
vars             x, y : entry   n : nat   a : acc   as : set(acc)   c : com
                 m, m′ : message
Horn axioms      as : open(x) ≡ insert(new(x), as)   ⇐   x ∉ as
                 as : open(x) ≡ as   ⇐   x ∈ as
                 as : close(x) ≡ remove(a, as)   ⇐   name(a) ≡ x
                 as : x.c ≡ insert(a : c, remove(a, as))
                          ⇐   name(a) ≡ x ∧ a ∈ as
                 as : x.c ≡ as   ⇐   x ∉ as
                 as : (m; m′) ≡ (as : m) : m′
                 transfer(x, n, y)  ≡  x.send(n, y); y.receive(n, x)
                 x ∈ as   ⇐   name(a) ≡ x ∧ a ∈ as
                 x ∉ as   ⇐   name(a) ≡ x ∧ a ∉ as
co-Horn axioms   m ≈ m′   ⇒   as : m ∼ as : m′
```

ACC_LOCAL2 = **ACC_STATE** `then`

`hidsorts`	com
`objconstructs`	$deposit, withdraw : nat \to com$
	$send, receive : nat \times entry \to com$
	$bal? : nat \to com$
`dynamic preds`	$\_ \xrightarrow{\_} \_ : acc \times com \times acc$
`vars`	$x : entry \quad n : nat \quad a : acc \quad c : com$
`Horn axioms`	$a \xrightarrow{deposit(n)} credit(a, n, name(a))$
	$a \xrightarrow{withdraw(n)} debit(a, n, name(a)) \quad \Leftarrow \quad n \leq bal(a)$
	$a \xrightarrow{send(n,x)} debit(a, n, x) \quad \Leftarrow \quad n \leq bal(a)$
	$a \xrightarrow{receive(n,x)} credit(a, n, x)$
	$a \xrightarrow{bal?(n)} a \quad \Leftarrow \quad n \equiv bal(a)$

$\mathbf{ACC\_GLOBAL2} = \mathbf{ACC\_LOCAL2}$ `and` $\mathbf{SET}$ `then`

`objconstructs`	$open, close : entry \to com$
	$\_.\_ : entry \times com \to com$
	$\_;\_ : com \times com \to com$
`defuncts`	$transfer : entry \times nat \times entry \to com$
`dynamic preds`	$\_ \Longrightarrow \_ : set(acc) \times com \times set(acc)$
`static preds`	$\_ \not\in \_ : entry \times set(acc)$
ν`-preds`	$\_ \approx \_ : com \times com$
`vars`	$x, y : entry \quad n : nat \quad a, a' : acc$
	$as, bs, as', bs' : set(acc) \quad c, c' : com$
`Horn axioms`	$as \stackrel{open(x)}{\Longrightarrow} insert(new(x), as) \quad \Leftarrow \quad x \not\in as$
	$as \stackrel{close(x)}{\Longrightarrow} remove(a, as) \quad \Leftarrow \quad name(a) \equiv x$
	$as \stackrel{x.c}{\Longrightarrow} insert(b, remove(a, as))$
	$\qquad \Leftarrow \quad name(a) \equiv x \wedge a \in as \wedge a \stackrel{c}{\longrightarrow} b$
	$as \stackrel{c;c'}{\Longrightarrow} as' \quad \Leftarrow \quad as \stackrel{c}{\Longrightarrow} bs \wedge bs \stackrel{c'}{\Longrightarrow} as'$
	$transfer(x, n, y) \;\equiv\; x.send(n, y);\; y.receive(n, x)$
	$x \not\in as \quad \Leftarrow \quad name(a) \equiv x \wedge a \not\in as$
`co-Horn axioms`	$c \approx c' \quad \Rightarrow \quad (as \stackrel{c}{\Longrightarrow} bs$
	$\qquad \Rightarrow \; \exists\, bs' : (as \stackrel{c'}{\Longrightarrow} bs' \wedge bs \sim bs'))$
	$c \approx c' \quad \Rightarrow \quad (as \stackrel{c'}{\Longrightarrow} bs'$
	$\qquad \Rightarrow \; \exists\, bs : (as \stackrel{c}{\Longrightarrow} bs \wedge bs \sim bs'))$

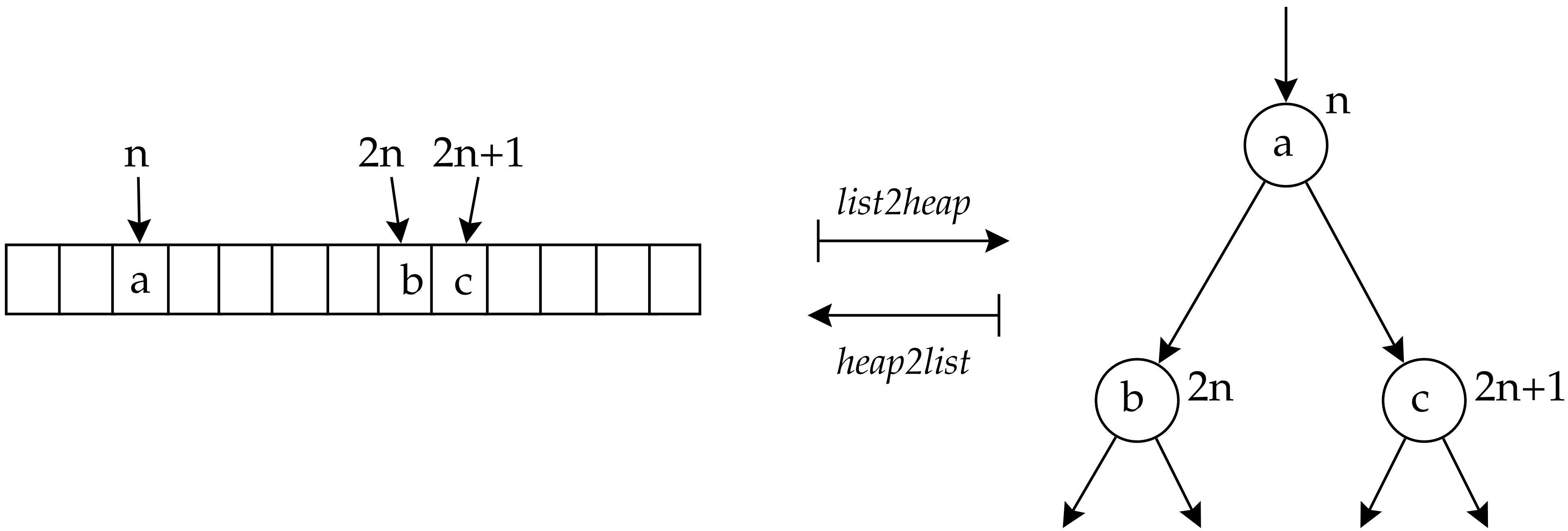
ACC_GLOBAL3 = ACC_LOCAL2 `and` **LIST** `and` **MAP** `then`

`hidsorts`	$accs = map(entry, list(com))$
`objconstructs`	$open, close : entry \to com$
	$\_.\_ : entry \times com \to com$
`defuncts`	$transfer : entry \times nat \times entry \to com$
`static preds`	$A : list(com) \times acc$
	$AS : list(com) \times accs$
	$free : list(com)$
`dynamic preds`	$\_ \Longrightarrow \_ : set(acc) \times com \times set(acc)$
ν`-preds`	$\_ \approx \_ : list(com) \times list(com)$
`vars`	$x, y : entry \quad c, c' : com \quad cL, cL' : list(com)$
	$a, a' : acc \quad as, bs, as' : accs$
`Horn axioms`	$A(nil, a)$
	$A(c :: cL, a) \quad \Leftarrow \quad a \xrightarrow{c} a' \wedge A(cL, a')$
	$AS(nil, as)$
	$AS(m :: mL, as) \quad \Leftarrow \quad as \stackrel{m}{\Longrightarrow} as' \wedge AS(mL, as')$
	$as \stackrel{open(x)}{\Longrightarrow} upd(x, cL, as)$
	$\Leftarrow \quad get(as, x) \equiv () \wedge free(cL) \wedge A(cL, new(x))$
	A port cL is created and assigned to x. Commands arriving at cL are processed by A in state $new(x)$.
	$as \stackrel{close(x)}{\Longrightarrow} remove(x, as)$
	$as \stackrel{x.c}{\Longrightarrow} as \quad \Leftarrow \quad get(as, x) \equiv (cL) \wedge c \in cL$
	Command c is sent to cL, the port of x.
	$transfer(x, n, y)$
	$\equiv \quad x.send(n, y) :: y.receive(n, x) :: nil$
`co-Horn axioms`	$cL \approx cL'$
	$\Rightarrow \quad (AS(cL, as) \Rightarrow \exists\, as' : (AS(cL', as') \wedge as \sim as'))$
	$cL \approx cL'$
	$\Rightarrow \quad (AS(cL', as') \Rightarrow \exists\, as : (AS(cL, as) \wedge as \sim as'))$

HEAP $=$ **LIST** `then`

`hidsorts`	$bintree = bintree(entry)$
`objconstructs`	$mt\!:\to bintree$
	$\_\#\_\#\_\!: bintree \times entry \times bintree \to bintree$
`defuncts`	$list2heap\!: list \to bintree$
	$mkHeap\!: list \times nat \to 1 + bintree$
`vars`	$x : entry \quad n : nat \quad L : list \quad T, T' : bintree$
`axioms`	$list2heap(nil) \equiv mt$

$$
\begin{aligned}
& list2heap(x :: L) \equiv T \\
& \qquad\qquad \Leftarrow \quad mkHeap(x :: L, 1) \equiv (T) \\
& mkHeap(L, n) \equiv () \quad \Leftarrow \quad nth(L, n-1) \equiv () \\
& mkHeap(L, n) \equiv (mt\#x\#mt) \\
& \qquad\qquad \Leftarrow \quad nth(L, n-1) \equiv (x) \ \wedge \\
& \qquad\qquad\qquad mkHeap(L, 2*n) \equiv () \ \wedge \\
& \qquad\qquad\qquad mkHeap(L, 2*n+1) \equiv () \\
& mkHeap(L, n) \equiv (mt\#x\#T) \\
& \qquad\qquad \Leftarrow \quad nth(L, n-1) \equiv (x) \ \wedge \\
& \qquad\qquad\qquad mkHeap(L, 2*n) \equiv () \ \wedge \\
& \qquad\qquad\qquad mkHeap(L, 2*n+1) \equiv (T) \\
& mkHeap(L, n) \equiv (T\#x\#mt) \\
& \qquad\qquad \Leftarrow \quad nth(L, n-1) \equiv (x) \ \wedge \\
& \qquad\qquad\qquad mkHeap(L, 2*n) \equiv (T) \ \wedge \\
& \qquad\qquad\qquad mkHeap(L, 2*n+1) \equiv () \\
& mkHeap(L, n) \equiv (T\#x\#T') \\
& \qquad\qquad \Leftarrow \quad nth(L, n-1) \equiv (x) \ \wedge \\
& \qquad\qquad\qquad mkHeap(L, 2*n) \equiv (T) \ \wedge \\
& \qquad\qquad\qquad mkHeap(L, 2*n+1) \equiv (T')
\end{aligned}
$$


n
2n
2n+1
a
b
c
list2heap
heap2list
n
a
b
2n
c
2n+1

FINSET = LIST `then`

`hidsorts`	$set = set(entry) \quad set' = set(entry')$
`constructs`	$\emptyset : \to set$
	$\{\_\} : entry \to set$
	$\_ \cup \_ : set \times set \to set$
`destructs`	$in : entry \times set \to bool$
`defuncts`	$mkset : 1 + \amalg_{n>0} entry^n \to set$
	$\_ \setminus \_ : set \times set \to set$
	$remove : entry \times set \to set$
	$map : (entry \to entry') \times set \to set'$
	$filter : (entry \to bool) \times set \to set$
	$forall : (entry \to bool) \times set \to bool$
	$\lvert\_\rvert : set \to nat$
	$flatten : set(set) \to set$
`vars`	$x, y, x_1, \ldots, x_n : entry \quad s, s' : set \quad S, S' : set(set)$
	$f : entry \to entry' \quad g : entry \to bool$
`Horn axioms`	$in(x, \emptyset) \equiv false$
	$in(x, \{y\}) \equiv eq(x, y)$
	$in(x, s \cup s') \equiv in(x, s) \ or \ in(x, s')$
	$mkset(()) \equiv \emptyset$
	$mkset((x_1, \ldots, x_n)) \equiv \{x_1\} \cup \cdots \cup \{x_n\}$
	$s \setminus s' \equiv filter(\lambda x.not(in(x, s')), s)$
	$remove(x, s) \equiv s \setminus \{x\}$
	$map(f, \emptyset) \equiv \emptyset$
	$map(f, \{x\}) \equiv \{f(x)\}$
	$map(f, s \cup s') \equiv map(f, s) \cup map(f, s')$
	$filter(g, \emptyset) \equiv \emptyset$
	$filter(g, \{x\}) \equiv \emptyset \Leftarrow g(x) \equiv false$
	$filter(g, \{x\}) \equiv \{x\} \Leftarrow g(x) \equiv true$
	$filter(g, s \cup s') \equiv filter(g, s) \cup filter(g, s')$
	$forall(g, \emptyset) \equiv true$
	$forall(g, \{x\}) \equiv g(x)$
	$forall(g, s \cup s') \equiv forall(g, s) \ and \ forall(g, s')$

$$
\begin{aligned}
|\emptyset| &\equiv 0 \\
|\{x\}| &\equiv 1 \\
|s \cup s'| &\equiv |s \setminus s'| + |s' \setminus s|
\end{aligned}
$$

$$flatten(\emptyset) \equiv \emptyset$$

$$flatten(\{S\}) \equiv S$$

$$flatten(S \cup S') \equiv flatten(S) \cup flatten(S')$$

LIST2SET = ORDER($entry$) `and` **ORDER**($entry'$) `and` **NAT** `then`

`hidsorts`	$set = set(entry) \quad set' = set(entry')$
`objconstructs`	$nil : \rightarrow set$
	$\_ :: \_ : entry \times set \rightarrow set$
`expfuncts`	$\emptyset : \rightarrow set$
	$insert, remove : entry \times set \rightarrow set$
	$mkset : list \rightarrow set$
	$filter : (entry \rightarrow bool) \times set \rightarrow set$
	$map : (entry \rightarrow entry') \times set \rightarrow set'$
	$\lvert\_\rvert : set \rightarrow nat$
	$in : entry \times set \rightarrow bool$
	$exists, forall : (entry \rightarrow bool) \times set \rightarrow bool$
`vars`	$x, y : entry \quad s, s' : set \quad L : list$
	$f : entry \rightarrow entry' \quad g : entry \rightarrow bool$
`Horn axioms`	$\emptyset \;\equiv\; nil$
	$insert(x, nil) \;\equiv\; x :: nil$
	$insert(x, y :: s) \;\equiv\; x :: s$
	$insert(x, y :: s) \;\equiv\; x :: (y :: s) \quad \Leftarrow \quad x < y$
	$insert(x, y :: s) \;\equiv\; y :: insert(x, s) \quad \Leftarrow \quad x > y$
	$remove(x, s) \;\equiv\; filter(\lambda y.not(eq(x, y)), s)$
	$mkset(nil) \;\equiv\; \emptyset$
	$mkset(x :: L) \;\equiv\; insert(x, mkset(L))$
	$filter(g, nil) \;\equiv\; nil$
	$filter(g, x :: s) \;\equiv\; filter(g, s) \quad \Leftarrow \quad g(x) \equiv false$
	$filter(g, x :: s) \;\equiv\; x :: filter(g, s) \quad \Leftarrow \quad g(x) \equiv true$
	$map(f, nil) \;\equiv\; nil$
	$map(f, x :: s) \;\equiv\; insert(f(x), map(f, s))$
	$\lvert nil \rvert \;\equiv\; 0$
	$\lvert x :: s \rvert \;\equiv\; \lvert s \rvert + 1$
	$in(x, s) \;\equiv\; exists(\lambda y.eq(x, y), s)$
	$exists(g, nil) \;\equiv\; false$
	$exists(g, x :: s) \;\equiv\; (g(x)\ or\ exists(g, s))$
	$forall(g, s) \;\equiv\; not(exists(not \circ g, s))$

INFSEQ = **LIST** `and` **NAT** `then`

`hidsorts`	$infseq = infseq(entry)$
`constructs`	$\_\&\_ : entry \times infseq \to infseq$
	$blink : \to infseq(nat)$
	$nats : nat \to infseq$
	$odds : infseq \to infseq$
	$zip : infseq \times infseq \to infseq$
	$map : (entry \to entry) \times infseq \to infseq$
`destructs`	$head : infseq \to entry$
	$tail : infseq \to infseq$
`defuncts`	$\_\#\_ : list \times infseq \to infseq$
	$evens : infseq \to infseq$
	$firstn : nat \times infseq \to list$
	$nth : nat \times infseq \to entry$
	$nthtail : nat \times infseq \to infseq$
`static preds`	$exists : (entry \to bool) \times infseq$
ν`-preds`	$forall, forallExists : (entry \to bool) \times infseq$
	$fair : (entry \to bool) \times infseq$
`vars`	$n : nat \quad x, y : entry \quad L : list \quad s, s' : infseq$
	$f : entry \to entry \quad g : entry \to bool$
`Horn axioms`	$head(x\&s) \equiv x$
	$head(blink) \equiv 0$
	$head(nats(n)) \equiv n$
	$head(zip(s, s')) \equiv head(s)$
	$head(odds(s)) \equiv head(s)$
	$head(map(f, s)) \equiv f(s)$
	$tail(x\&s) \equiv s$
	$tail(blink) \equiv 1\&blink$
	$tail(nats(n)) \equiv nats(n+1)$
	$tail(zip(s, s')) \equiv zip(s', tail(s))$
	$tail(odds(s)) \equiv odds(tail(tail(s)))$
	$tail(map(f, s)) \equiv map(f, tail(s))$
	$nil\#s \equiv s$

$$
\begin{array}{ll}
 & (x :: L)\#s \equiv x\&(L\#s) \\
 & evens(s) \equiv odds(tail(s)) \\
 & firstn(0,s) \equiv nil \\
 & firstn(n+1,s) \equiv head(s) :: firstn(n, tail(s)) \\
 & nth(0,s) \equiv head(s) \\
 & nth(n+1,s) \equiv nth(n, tail(s)) \\
 & nthtail(0,s) \equiv s \\
 & nthtail(n+1,s) \equiv nthtail(n, tail(s)) \\
 & exists(g,s) \;\Leftarrow\; g(head(s)) \equiv true \\
 & exists(g,s) \;\Leftarrow\; exists(g, tail(s)) \\
\texttt{co-Horn axioms} & forall(g,s) \\
 & \quad \Rightarrow\; g(head(s)) \equiv true \wedge forall(g, tail(s)) \\
 & forallExists(g,s) \\
 & \quad \Rightarrow\; exists(g,s) \wedge forallExists(g, tail(s)) \\
\textbf{(A)} & fair(g,s) \;\Rightarrow\; forallExists(g,s) \\
\textbf{(B)} & fair(g,s) \;\Rightarrow\; exists(g,s) \wedge fair(g, tail(s)) \\
\textbf{(C)} & fair(g,s) \;\Rightarrow\; exists(g, nthtail(n,s))
\end{array}
$$

The INFSEQ-contexts:

$$CT \;=\; \{head \cdot tail^n : infseq \to entry \mid n \in \mathbb{N}\}$$

The final CSP-model:

$$
\begin{aligned}
Fin_{infseq} &= \textstyle\prod_{d \in CT} C_{entry} \\
&= \textstyle\prod_{n \in \mathbb{N}} C_{entry} \;=\; [\mathbb{N} \longrightarrow C_{entry}]
\end{aligned}
$$

The domain completion of INFSEQ:

INFSEQ `then`

`Horn axioms` $head(f) \equiv f(0)$

$tail(f) \equiv \lambda n.f(n+1)$ for all $f : \mathbb{N} \to C_{entry}$

FAIR$(G : entry \to bool)$ = **INFSEQ** `then`

`destructors` $min : (entry \to bool) \times infseq \to 1 + nat$

`Horn axioms` axioms for min

`assertions` $\exists\ n, s' : (min(G, s) \equiv (n) \wedge nthtail(n+1, s) \equiv s')$

$min(g, s) \equiv () \quad \Rightarrow \quad g(nth(n, s)) \equiv false$

$min(g, s) \equiv (n) \quad \Rightarrow \quad g(nth(n, s) \equiv true$

$min(g, s) \equiv (n) \wedge m < n) \quad \Rightarrow \quad g(nth(m, s) \equiv false$

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathbf{FSTREAM} = \mathbf{ENTRY}(entry)\ \texttt{and}\ \mathbf{BOOL}\ \texttt{and}\ \mathbf{NAT}\ \texttt{then}} \\
\quad \texttt{hidsorts} & stream = stream(entry) \\
\quad \texttt{destructs} & ht\!:\!stream \to 1 + (entry \times stream) \\
\quad \texttt{constructs} & nil\!:\to stream \\
 & \_\&\_\!:\!entry \times stream \to stream \\
 & blink\!:\to stream \\
 & nats\!:\!nat \to stream \\
 & odds\!:\!stream \to stream \\
 & zip\!:\!stream \times stream \to stream \\
 & map\!:\!(entry \to entry) \times stream \to stream \\
\quad \texttt{defuncts} & evens\!:\!stream \to stream \\
 & firstn\!:\!nat \times stream \to stream \\
 & nth\!:\!nat \times stream \to 1 + entry \\
 & nthtail\!:\!nat \times stream \to stream \\
\quad \texttt{static preds} & isnil : stream \\
 & exists : (entry \to bool) \times stream \\
\quad \nu\texttt{-preds} & forall : (entry \to bool) \times stream \\
 & fair : (entry \to bool) \times stream \\
 & infinite : stream \\
\quad \texttt{vars} & n : nat \quad x, y : entry \quad s, s', t : stream \\
 & f\!:\!entry \to entry \quad g\!:\!entry \to bool \\
\quad \texttt{Horn axioms} & ht(nil) \equiv () \\
 & ht(x\&s) \equiv (x, s) \\
 & ht(blink) \equiv (0, 1\&blink) \\
 & ht(nats(n)) \equiv (n, nats(n+1)) \\
 & ht(zip(s, s')) \equiv ht(s') \quad \Leftarrow \quad ht(s) \equiv () \\
 & ht(zip(s, s')) \equiv (x, zip(s', t)) \quad \Leftarrow \quad ht(s) \equiv (x, t) \\
 & ht(odds(s)) \equiv () \quad \Leftarrow \quad ht(s) \equiv () \\
 & ht(odds(s)) \equiv (x, nil) \\
 & \qquad \Leftarrow \quad ht(s) \equiv (x, t) \wedge ht(t) \equiv () \\
 & ht(odds(s)) \equiv (x, odds(t)) \\
 & \qquad \Leftarrow \quad ht(s) \equiv (x, t) \wedge ht(t) \equiv (y, t) \\
 & ht(map(f, s)) \equiv () \quad \Leftarrow \quad ht(s) \equiv ()
\end{array}
$$

$$
\begin{aligned}
& ht(map(f,s)) \equiv (f(s), map(f,t)) \\
& \quad \Leftarrow \quad ht(s) \equiv (x,t) \\
& evens(s) \equiv () \quad \Leftarrow \quad ht(s) \equiv () \\
& evens(s) \equiv odds(t) \quad \Leftarrow \quad ht(s) \equiv (x,t) \\
& firstn(0,s) \equiv nil \\
& firstn(n+1,s) \equiv x \& firstn(n,t) \\
& \quad \Leftarrow \quad ht(s) \equiv (x,t) \\
& nth(n,s) \equiv () \quad \Leftarrow \quad ht(s) \equiv () \\
& nth(0,s) \equiv (x) \quad \Leftarrow \quad ht(s) \equiv (x,t) \\
& nth(n+1,s) \equiv nth(n,t) \quad \Leftarrow \quad ht(s) \equiv (x,t) \\
& nthtail(0,s) \equiv nil \quad \Leftarrow \quad ht(s) \equiv () \\
& nthtail(0,s) \equiv s \quad \Leftarrow \quad ht(s) \equiv (x,t) \\
& nthtail(n+1,s) \equiv nthtail(n,t) \quad \Leftarrow \quad ht(s) \equiv (x,t) \\
& isnil(s) \quad \Leftarrow \quad ht(s) \equiv () \\
& exists(g,s) \quad \Leftarrow \quad ht(s) \equiv (x,t) \wedge g(x) \equiv true \\
& exists(g,s) \quad \Leftarrow \quad ht(s) \equiv (x,t) \wedge exists(g,t)
\end{aligned}
$$

`co-Horn axioms`

$$
\begin{aligned}
& forall(g,s) \\
& \quad \Rightarrow \quad (ht(s) \equiv (x,t) \;\Rightarrow\; (g(x) \equiv true \wedge forall(g,t))) \\
& fair(g,s) \\
& \quad \Rightarrow \quad (ht(s) \equiv (x,t) \;\Rightarrow\; (exists(g,s) \wedge fair(g,t))) \\
& infinite(s) \\
& \quad \Rightarrow \quad \exists x, t : (ht(s) \equiv (x,t) \wedge infinite(t))
\end{aligned}
$$

FILTER1 = FSTREAM `then`

`defuncts`	$filter\!:\!(entry \to bool) \times stream \to stream$
`static preds`	$forall : (entry \to bool) \times stream$
	$forall' : nat \times (entry \to bool) \times stream$
	$r : nat \times (entry \to bool) \times stream \times entry \times stream$
`vars`	$i : nat \quad x : entry \quad s, t : stream$
	$g\!:\!entry \to bool$
`Horn axioms`	$forall(g, s) \quad \Leftarrow \quad \forall i : forall'(i, g, s)$
	$forall'(0, g, s)$
	$forall'(i + 1, g, s) \quad \Leftarrow \quad \forall x, t : r(i, g, s, x, t)$
	$r(i, g, s, x, t) \quad \Leftarrow \quad ht(s) \equiv ()$
	$r(i, g, s, x, t) \quad \Leftarrow \quad g(x) \equiv true \wedge forall'(i, g, t)$
	$filter(g, s) \equiv nil \quad \Leftarrow \quad forall(not \circ g, s)$
	$filter(g, s) \equiv x \& filter(g, t)$
	$\quad \Leftarrow \quad ht(s) \equiv (x, t) \wedge g(x) \equiv true$
	$filter(g, s) \equiv filter(g, t)$
	$\quad \Leftarrow \quad ht(s) \equiv (x, t) \wedge g(x) \equiv false \wedge exists(g, t)$

The FSTREAM-contexts:

$$
\begin{aligned}
CT \;=\; \{c(n) = {} & \boxed{(id+}^{n}(id+\pi_1)\cdot ht \cdot \boxed{\pi_2)\cdot ht}^{n} \\
& : stream \to \boxed{1+(}^{n} 1 + entry \boxed{)}^{n} \;\mid\; n \in \mathbb{N}\}
\end{aligned}
$$

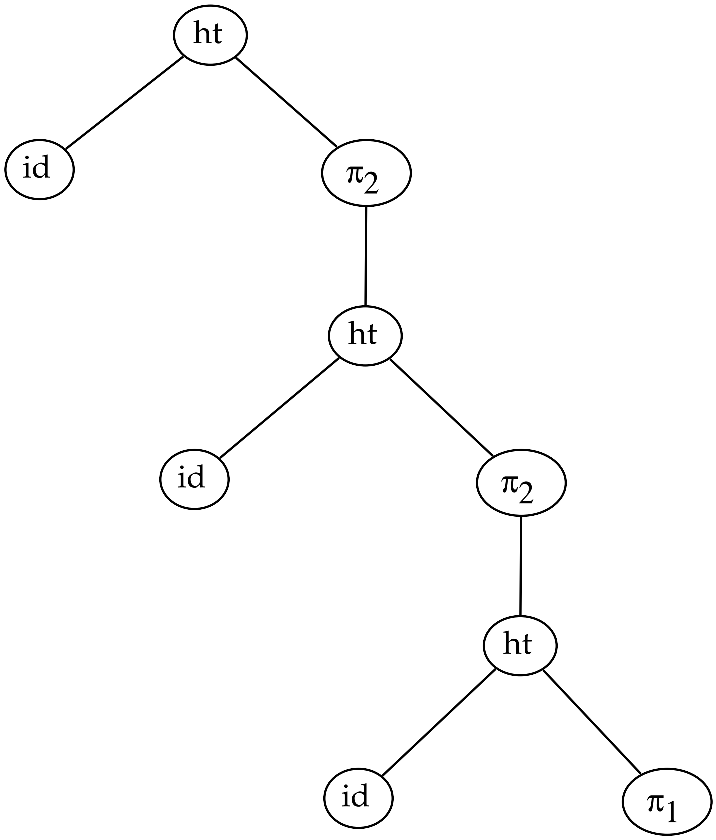
The final CSP-model:

$$
P \;=\; \prod_{n\in\mathbb{N}} \left(\boxed{1+(}^{n} 1 + C_{entry} \boxed{)}^{n}\right)
$$

$$
\begin{aligned}
Fin_{stream} \;=\; \{a \in P \mid {} & \forall\, n \in \mathbb{N} : \pi_{c(n+1)}(a) = () \;\vee \\
& \exists\, a' \in Fin_{stream} \\
& \quad \forall\, n \in \mathbb{N} : \pi_{c(n+1)}(a) = (\pi_{c(n)}(a'))\}
\end{aligned}
$$

$$
= \; 1 + C_{entry} \times (1 + C_{entry} \times \ldots)
$$

$$
= \; C^{*}_{entry} \cup [\mathbb{N} \longrightarrow C_{entry}]
$$



The context

$$
\begin{aligned}
c(2) \;&=\; (id + (id + (id + \pi_1) \cdot ht \cdot \pi_2) \cdot ht \cdot \pi_2) \cdot ht \\
&\quad : stream \rightarrow 1 + (1 + (1 + entry))
\end{aligned}
$$

The domain completion of FSTREAM:

FSTREAM `then`

`Horn axioms` $ht(nil) \equiv ()$

$ht(x :: L) \equiv (x, L)$ for all $x \in C_{entry}$ and $L \in C^*_{entry}$

$ht(f) \equiv (f(0), \lambda n.f(n+1))$ for all $f : \mathbb{N} \to C_{entry}$

INFINITE $=$ **FSTREAM** `then`

`assertions` $\exists\, x, s' : ht(s) \equiv (x, s')$

CONSTANT $=$ **FSTREAM** `then`

`assertions` $\exists\, x, s_1, s_2 : ht(s) \equiv (x, s_1) \wedge ht(s_1) \equiv (x, s_2)$

ALTERNATING $=$ **FSTREAM** `then`

`assertions`
$$\exists\, x, s_1, s_2, s_3 : (ht(s) \equiv (x, s_1) \wedge ht(s_1) \equiv (y, s_2) \wedge ht(s_2) \equiv (x, s_3)) \vee$$
$$\exists\, x, s_1, s_2, s_3 : (ht(s) \equiv (y, s_1) \wedge ht(s_1) \equiv (x, s_2) \wedge ht(s_2) \equiv (y, s_3))$$

FILTER = FSTREAM `then`

`destructors`	$min\colon(entry \to bool) \times stream \to 1 + nat$
`Horn axioms`	axioms for min
`cofuncts`	$odds\colon stream \to stream$
	$filter\colon(entry \to bool) \times stream \to stream$
`vars`	$m, n : nat \quad s, t : stream \quad g\colon entry \to bool$
`Horn axioms`	$ht(odds(s)) \equiv () \quad \Leftarrow \quad ht(s) \equiv ()$
	$ht(odds(s)) \equiv (x, nil)$
	$\quad \Leftarrow \quad ht(s) \equiv (x, t) \wedge ht(t) \equiv ()$
	$ht(odds(s)) \equiv (x, odds(s'))$
	$\quad \Leftarrow \quad ht(s) \equiv (x, t) \wedge ht(t) \equiv (y, s')$
	$ht(filter(g, s)) \equiv () \quad \Leftarrow \quad min(g, s) \equiv ()$
	$ht(filter(g, s)) \equiv (x, filter(g, nthtail(n + 1, s)))$
	$\quad \Leftarrow \quad min(g, s) \equiv (n) \wedge nth(n, s) \equiv (x)$
`assertions`	$min(g, s) \equiv () \quad \Rightarrow \quad g(nth(n, s)) \equiv false$
	$min(g, s) \equiv (n) \quad \Rightarrow \quad g(nth(n, s) \equiv true$
	$(min(g, s) \equiv (n) \wedge m < n \quad \Rightarrow \quad g(nth(m, s) \equiv false$

STREAM $=$ **LIST** `and` **NAT** `then`

`hidsorts`	$stream$
`constructs`	$nil :\rightarrow stream$
	$\_\&\_ : entry \times stream \rightarrow stream$
	$blink :\rightarrow stream(nat)$
	$nats : nat \rightarrow stream$
	$odds, evens : stream \rightarrow stream$
	$zip : stream \times stream \rightarrow stream$
	$map : (entry \rightarrow entry) \times stream \rightarrow stream$
	$filter : (entry \rightarrow bool) \times stream \rightarrow stream$
`defuncts`	$firstn : nat \times stream \rightarrow list$
	$nthtail : nat \times stream \rightarrow stream$
`destructors`	$isnil : stream$
`transpreds`	$\_ \xrightarrow{\_} \_ : stream \times entry \times stream$
`static preds`	$isnil, finite : stream$
	$exists : (entry \rightarrow bool) \times stream$
ν`-preds`	$forall, forallExists : (entry \rightarrow bool) \times stream$
	$fair : (entry \rightarrow bool) \times stream$
	$infinite : stream$
`vars`	$n : nat \quad x, y : entry \quad L : list \quad s, s', t, t' : stream$
	$f : entry \rightarrow entry \quad g : entry \rightarrow bool$
`Horn axioms`	$x\&s \xrightarrow{x} s$
	$blink \xrightarrow{0} 1\&blink$
	$nats(n) \xrightarrow{n} nats(n+1)$
	$odds(s) \xrightarrow{x} odds(t) \;\Leftarrow\; s \xrightarrow{x} s' \wedge s' \xrightarrow{y} t$
	$evens(s) \xrightarrow{x} odds(t) \;\Leftarrow\; s \xrightarrow{y} s' \wedge s' \xrightarrow{x} t$
	$zip(s, s') \xrightarrow{x} zip(s', t) \;\Leftarrow\; s \xrightarrow{x} t$
	$zip(s, s') \xrightarrow{x} zip(s, t) \;\Leftarrow\; isnil(s) \wedge s' \xrightarrow{x} t$
	$map(f, s) \xrightarrow{f(x)} map(f, t) \;\Leftarrow\; s \xrightarrow{x} t$

$$
\begin{array}{ll}
 & filter(g,s) \xrightarrow{x} filter(g,t) \\
 & \quad \Leftarrow \;\; s \xrightarrow{x} t \wedge g(x) \equiv true \\
 & filter(g,s) \xrightarrow{y} t' \\
 & \quad \Leftarrow \;\; s \xrightarrow{x} t \wedge g(x) \equiv false \wedge filter(g,t) \xrightarrow{y} t' \\
 & isnil(nil) \\
 & isnil(odds(s)) \;\; \Leftarrow \;\; isnil(s) \\
 & isnil(evens(s)) \;\; \Leftarrow \;\; isnil(s) \\
 & isnil(evens(s)) \quad \Leftarrow \quad s \xrightarrow{x} t \wedge isnil(t) \\
 & isnil(zip(s,s')) \;\; \Leftarrow \;\; isnil(s) \wedge isnil(s') \\
 & isnil(map(f,s)) \;\; \Leftarrow \;\; isnil(s) \\
 & isnil(filter(g,s)) \;\; \Leftarrow \;\; isnil(s) \\
 & isnil(filter(g,s)) \\
 & \quad \Leftarrow \;\; s \xrightarrow{x} t \wedge g(x) \equiv false \wedge isnil(filter(g,t)) \\
 & finite(s) \;\; \Leftarrow \;\; isnil(s) \\
 & finite(s) \;\; \Leftarrow \;\; s \xrightarrow{x} t \wedge finite(t) \\
 & exists(g,s) \;\; \Leftarrow \;\; s \xrightarrow{x} t \wedge g(x) \equiv true \\
 & exists(g,s) \;\; \Leftarrow \;\; s \xrightarrow{x} t \wedge exists(g,t) \\
 & firstn(0,s) \equiv nil \\
 & firstn(n+1,s) \equiv x :: firstn(n,t) \quad \Leftarrow \quad s \xrightarrow{x} t \\
 & nthtail(0,s) \equiv s \\
 & nthtail(n+1,s) \equiv nthtail(n,t) \quad \Leftarrow \quad s \xrightarrow{x} t \\
\texttt{co-Horn axioms} & forall(g,s) \\
 & \quad \Rightarrow \;\; (s \xrightarrow{x} t \;\; \Rightarrow \;\; (g(x) \equiv true \wedge forall(g,t))) \\
 & forallExists(g,s) \;\; \Rightarrow \;\; exists(g,s) \\
 & forallExists(g,s) \\
 & \quad \Rightarrow \;\; (s \xrightarrow{x} t \;\; \Rightarrow \;\; forallExists(g,t)) \\
\textbf{(A)} & fair(g,s) \;\; \Rightarrow \;\; forallExists(g,s) \\
\textbf{(B1)} & fair(g,s) \;\; \Rightarrow \;\; exists(g,s) \\
\textbf{(B2)} & fair(g,s) \;\; \Rightarrow \;\; (s \xrightarrow{x} t \;\; \Rightarrow \;\; fair(g,t)) \\
 & infinite(s) \;\; \Rightarrow \;\; \exists x,t : (s \xrightarrow{x} t \wedge infinite(t))
\end{array}
$$

$\mathbf{FPROCESS} = \mathbf{ACTION}$ `and` $\mathbf{BOOL}$ `then`

`hidsorts`	$proc \quad procs$
`constructs`	$\_.\_ : act \times procs \rightarrow proc$
	$\emptyset : \rightarrow procs$
	$\{\_\} : proc \rightarrow procs$
	$\_ + \_ : procs \times procs \rightarrow procs$
	$\_;\_ : procs \times procs \rightarrow procs$
	$\_\vert\_ : procs \times procs \rightarrow procs$
	$map : (act \rightarrow act) \times proc \rightarrow proc$
	$map : (act \rightarrow act) \times procs \rightarrow procs$
`destructs`	$root : proc \rightarrow act$
	$subs : proc \rightarrow procs$
`transpreds`	$\_ \ni \_ : procs \times proc$
`vars`	$a : act \quad p, q : proc \quad ps, ps', qs, qs' : procs \quad f : act \rightarrow act$
`Horn axioms`	$root(a.ps) \equiv a$
	$subs(a.ps) \equiv ps$
	$\{p\} \ni p$
	$ps + qs \ni p \quad \Leftarrow \quad ps \ni p$
	$ps + qs \ni p \quad \Leftarrow \quad qs \ni p$
	$ps; qs \ni a.(ps'; qs)$
	$\quad \Leftarrow \quad ps \ni p \wedge root(p) \equiv a \wedge subs(p) \equiv ps'$
	$ps\vert qs \ni a.(ps'\vert qs)$
	$\quad \Leftarrow \quad ps \ni p \wedge root(p) \equiv a \wedge subs(p) \equiv ps'$
	$ps\vert qs \ni a.(ps\vert qs')$
	$\quad \Leftarrow \quad qs \ni q \wedge root(q) \equiv a \wedge subs(q) \equiv qs'$
	$ps\vert qs \ni \tau.(ps'\vert qs')$
	$\quad \Leftarrow \quad ps \ni p \wedge root(p) \equiv a \wedge subs(p) \equiv ps' \wedge$
	$\quad \quad qs \ni q \wedge root(q) \equiv b \wedge subs(q) \equiv qs' \wedge \overline{a} \equiv b$
	$root(map(f,p)) \equiv f(root(p))$
	$subs(map(f,p)) \equiv map(f, subs(p))$
	$map(f, ps) \ni map(f,p) \quad \Leftarrow \quad ps \ni p$

FTREE = ENTRY$(entry)$ `then`

`hidsorts`	$tree = tree(entry)$	
`constructs`	$mt :\to tree$	
	$\_\&\_ : entry \times tree^n \to tree$	for all $n > 0$
`destructs`	$rs : tree \to 1 + (entry \times \coprod_{n>0} tree^n)$	
`vars`	$x : entry \quad t_1, \ldots, t_n : tree$	
`Horn axioms`	$rs(mt) \equiv ()$	
	$rs(x\&(t_1, \ldots, t_n)) \equiv (x, t_1, \ldots, t_n)$	

$$\mathbf{ETREE} = \mathbf{ENTRY}(entry)\ \texttt{then}$$

`hidsorts`	$etree = etree(entry)$
`constructs`	$mt\!:\to etree$
	$\_.\_\!: entry \times etree \to etree$
	$\_+\_\!: etree \times etree \to etree$
`destructs`	$sucs\!: etree \times entry \to 1 + \amalg_{n>0} etree^n$
`vars`	$x : entry \quad t, t', t_1, \ldots, t_m, t'_1, \ldots, t'_n : etree$
`Horn axioms`	$sucs(mt, x) \equiv ()$
	$sucs(x.t, x) \equiv (t)$
	$sucs(x.t, y) \equiv () \quad \Leftarrow \quad x \not\equiv y$
	$sucs(x, t + t') \equiv (t_1, \ldots, t_m, t'_1, \ldots, t'_n)$
	$\quad \Leftarrow \quad sucs(x, t) \equiv (t_1, \ldots, t_m) \ \wedge$
	$\qquad\qquad sucs(x, t') \equiv (t'_1, \ldots, t'_n)$
	$sucs(x, t + t') \equiv sucs(t) \quad \Leftarrow \quad sucs(x, t') \equiv ()$
	$sucs(x, t + t') \equiv sucs(t') \quad \Leftarrow \quad sucs(x, t) \equiv ()$

EPROCESS = ETREE `then`

`hidsorts`	$eproc = eproc(entry)$
`constructs`	$mkproc\colon etree \to eproc$
`defuncts`	$mt\colon \to eproc$
	$\_.\_\colon entry \times eproc \to eproc$
	$\_ + \_\colon eproc \times eproc \to eproc$
`transpreds`	$\_ \stackrel{\_}{\longrightarrow} \_ : eproc \times entry \times eproc$
`vars`	$x : entry \quad t, t_1, \ldots, t_n : etree \quad p, p' : eproc$
`Horn axioms`	$mkproc(t) \stackrel{x}{\longrightarrow} mkproc(t_i)$
	$\Leftarrow \quad sucs(t,x) \equiv (t_1, \ldots, t_n)$
	for all $1 \leq i \leq n$
	$mt \equiv mkproc(mt)$
	$x.mkproc(t) \equiv mkproc(x.t)$
	$mkproc(t) + mkproc(t') \equiv mkproc(t + t')$

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathbf{BINTREE} = \mathbf{ENTRY}(entry)\ \texttt{and}\ \mathbf{LIST}\ \texttt{then}} \\
\quad\texttt{hidsorts} & tree = tree(entry) \\
\quad\texttt{constructs} & mirror : tree \to tree \\
 & leaf, fill : entry \to tree \\
 & mkTree : tree \times entry \times tree \to tree \\
\quad\texttt{destructs} & root : tree \to entry \\
 & sucs : tree \to 1 + (tree \times tree) \\
 & size : tree \to 1 + nat \\
\quad\texttt{defuncts} & subtree : tree \times list(2) \to 1 + tree \\
\quad\texttt{static preds} & \_ \in \_ : entry \times tree \\
 & finite : tree \\
 & \Diamond r : tree \qquad\qquad \text{for all static } \mu\text{-predicates } r : tree \\
\quad\nu\texttt{-preds} & infinite : tree \\
 & \Box r : tree \qquad\qquad \text{for all static } \mu\text{-predicates } r : tree \\
\quad\texttt{vars} & x : entry \quad T, T', T_1, T_2 : tree \quad m, n : nat \quad b : bool \\
\quad\texttt{Horn axioms} & root(mirror(T)) \equiv root(T) \\
 & sucs(mirror(T)) \equiv () \quad \Leftarrow \quad sucs(T) \equiv () \\
 & sucs(mirror(T)) \equiv (mirror(T_2), mirror(T_1)) \\
 & \qquad\qquad \Leftarrow \quad sucs(T) \equiv (T_1, T_2) \\
 & size(mirror(T)) \equiv size(T) \\
 & root(leaf(x)) \equiv x \\
 & sucs(leaf(x)) \equiv () \\
 & size(leaf(x)) \equiv (1) \\
 & root(fill(x)) \equiv x \\
 & sucs(fill(x)) \equiv (fill(x), fill(x)) \\
 & size(fill(x)) \equiv () \\
 & root(mkTree(T_1, x, T_2)) \equiv x \\
 & sucs(mkTree(T_1, x, T_2)) \equiv (T_1, T_2) \\
 & size(mkTree(T_1, x, T_2)) \equiv () \\
 & \qquad\qquad \Leftarrow \quad size(T_1) \equiv () \\
 & size(mkTree(T_1, x, T_2)) \equiv () \\
 & \qquad\qquad \Leftarrow \quad size(T_2) \equiv () \\
 & size(mkTree(T_1, x, T_2)) \equiv (m + n + 1)
\end{array}
$$

$$
\begin{array}{l}
\qquad\qquad \Leftarrow \quad size(T_1) \equiv (m) \ \wedge \ size(T_2) \equiv (n) \\
subtree(T, nil) \equiv (T) \\
subtree(T, b :: L) \equiv () \\
\qquad\qquad \Leftarrow \quad subtree(T, L) \equiv () \\
subtree(T, b :: L) \equiv () \\
\qquad\qquad \Leftarrow \quad subtree(T, L) \equiv (T') \ \wedge \\
\qquad\qquad\qquad sucs(T') \equiv () \\
subtree(T, 0 :: L) \equiv (T_1) \\
\qquad\qquad \Leftarrow \quad subtree(T, L) \equiv (T') \ \wedge \\
\qquad\qquad\qquad sucs(T') \equiv (T_1, T_2) \\
subtree(T, 1 :: L) \equiv (T_2) \\
\qquad\qquad \Leftarrow \quad subtree(T, L) \equiv (T') \ \wedge \\
\qquad\qquad\qquad sucs(T') \equiv (T_1, T_2) \\
x \in T \quad \Leftarrow \quad root(T) \equiv x \\
x \in T \quad \Leftarrow \quad sucs(T) \equiv (T_1, T_2) \wedge x \in T_1 \\
x \in T \quad \Leftarrow \quad sucs(T) \equiv (T_1, T_2) \wedge x \in T_2 \\
finite(T) \quad \Leftarrow \quad sucs(T) \equiv () \\
finite(T) \quad \Leftarrow \quad sucs(T) \equiv (T_1, T_2) \ \wedge \\
\qquad\qquad\qquad\quad finite(T_1) \wedge finite(T_2) \\
\Diamond r(T) \quad \Leftarrow \quad r(T) \\
\Diamond r(T) \quad \Leftarrow \quad sucs(T) \equiv (T_1, T_2) \wedge \Diamond r(T_1) \\
\Diamond r(T) \quad \Leftarrow \quad sucs(T) \equiv (T_1, T_2) \wedge \Diamond r(T_2)
\end{array}
$$

`assertions` $sucs(T) \equiv () \Rightarrow size(T) \equiv (1)$

$$(sucs(T) \equiv (T_1, T_2) \wedge size(T_1) \equiv ()) \;\Rightarrow\; size(T) \equiv ()$$

$$(sucs(T) \equiv (T_1, T_2) \wedge size(T_2) \equiv ()) \;\Rightarrow\; size(T) \equiv ()$$

$$(sucs(T) \equiv (T_1, T_2) \wedge size(T_1) \equiv (m) \wedge size(T_2) \equiv (n)) \;\Rightarrow\; size(T) \equiv (m + n + 1)$$

`co-Horn axioms`

$$infinite(T) \;\Rightarrow\; (sucs(T) \equiv () \;\Rightarrow\; FALSE)$$

$$infinite(T) \;\Rightarrow\; (sucs(T) \equiv (T_1, T_2) \;\Rightarrow\; infinite(T_1) \vee infinite(T_2))$$

$$\Box r(T) \;\Rightarrow\; r(T)$$

$$\Box r(T) \;\Rightarrow\; (sucs(T) \equiv (T_1, T_2) \;\Rightarrow\; \Box r(T_1) \wedge \Box r(T_2))$$

The BINTREE-contexts:

$$\begin{aligned} d{:}tree \to entry \in CT &\implies d \cdot \pi_1, d \cdot \pi_2{:}tree \times tree \to entry \in CT, \\ d{:}tree \times tree \to entry \in CT &\implies (id + d) \cdot sucs{:}tree \to 1 + entry \in CT. \end{aligned}$$

The final CSP-model:

$$P \;=\; \prod_{c{:}tree \to s \in CT} C_s$$

$$\begin{aligned} Fin_{tree} \;=\; \{a \in P \mid \; & \exists\, b \in Fin_{tree \times tree} \\ & \forall\, d = (id + e) \cdot sucs \in \mathbb{N} : \pi_d(a) = (\pi_e(b)) \\ & \wedge\, Fin_{tree} \models_{a/T} AX_{MP}\} \end{aligned}$$

The domain completion of BINTREE:

BINTREE `then`

`Horn axioms` $root(f) \equiv f(\varepsilon)$

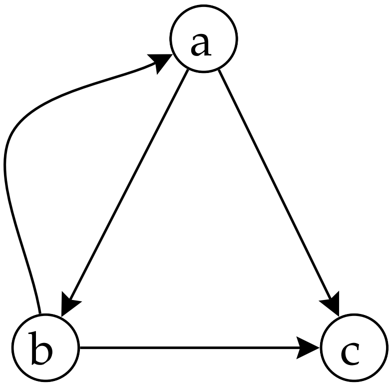
$$sucs(f) \equiv \begin{cases} (\lambda w.f(0w), \lambda w.f(1w)) & \\ & \text{if } f(0) \text{ and } f(1) \text{ are defined} \\ () & \text{otherwise} \end{cases}$$

$$size(f) \equiv \begin{cases} (|domain(f)|) & \text{if } domain(f) \text{ is finite} \\ () & \text{otherwise} \end{cases}$$

for all partial functions $f : 2^* \to C_{entry}$
that represent binary trees

BINTREE is membership compatible because for all axioms $is_{tree}(T) \Rightarrow \varphi$ of BINTREE and $t \in NF_{\Sigma',tree}$, $Her(SP') \models \varphi[t/T]$.

Any CST without hidS-constructors and axioms for hidS-destructors is membership compatible.



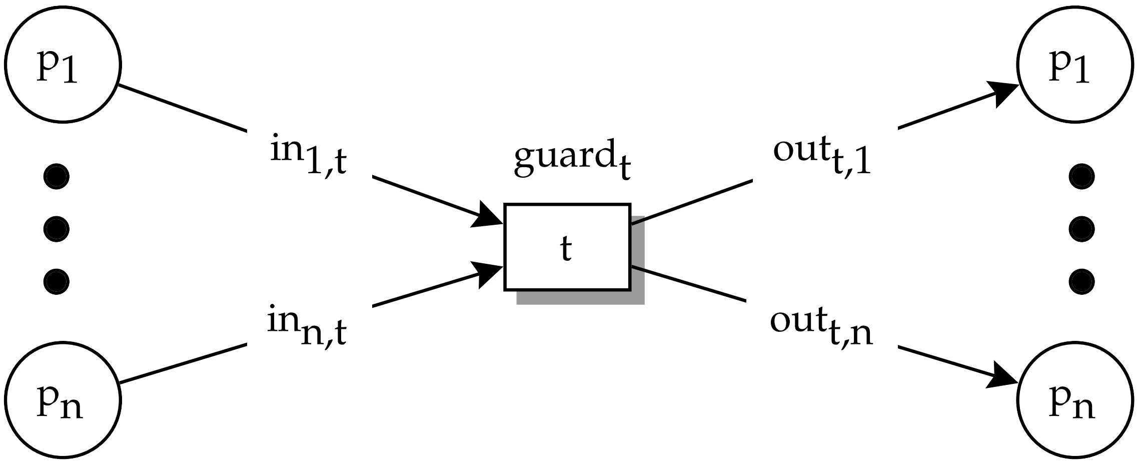
A binary graph

A_GRAPH = BINTREE `then`

`constructs`	$G_1, G_2, G_3 :\to tree$	
	$a, b, c :\to entry$	
`defuncts`	$G_3 :\to tree$	
`Horn axioms`	$root(G_1) \equiv a$	$sucs(G_1) \equiv (G_2, G_3)$
	$size(G_1) \equiv ()$	
	$root(G_2) \equiv b$	$sucs(G_2) \equiv (G_1, G_3)$
	$size(G_2) \equiv ()$	
	$root(G_3) \equiv c$	$sucs(G_2) \equiv ()$
	$size(G_3) \equiv (1)$	

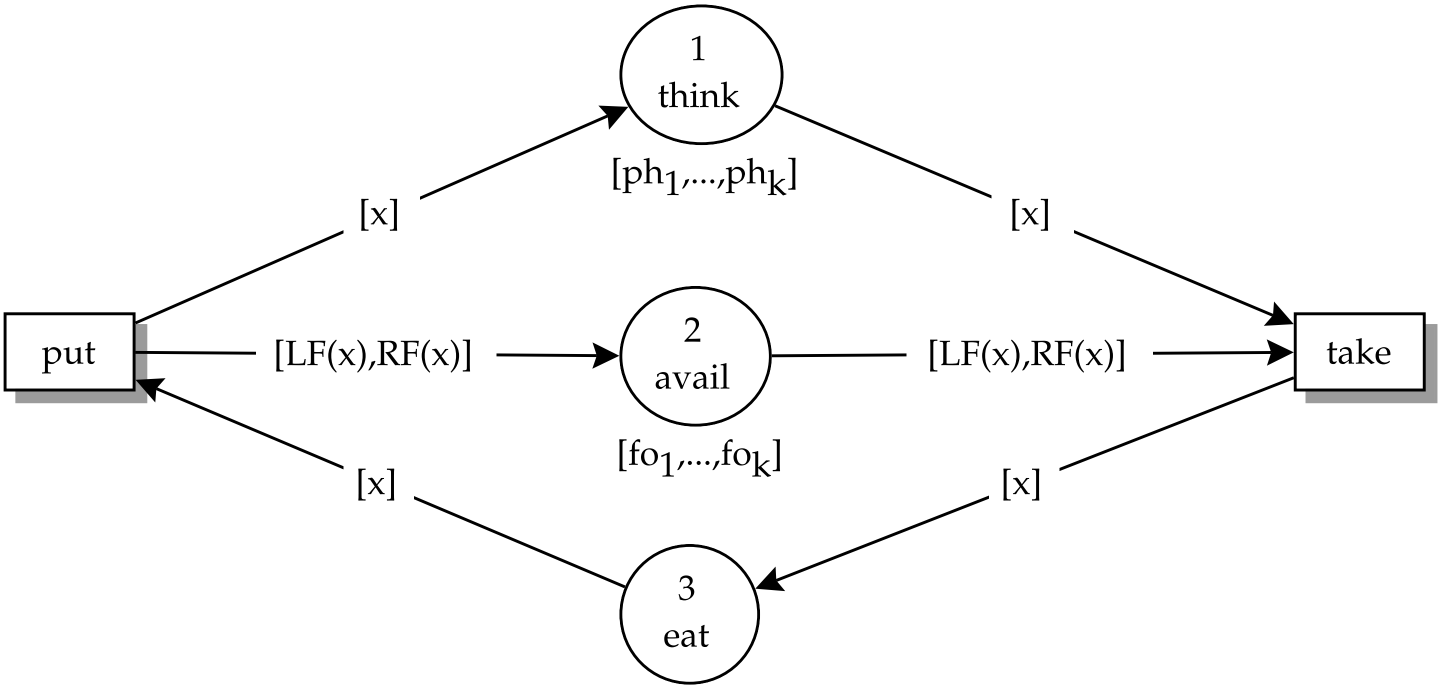
Algebraic/Colored/High-level Petri nets

Let N be a net with places $p_1,\dots,p_n$ and transitions $t_1,\dots,t_k$. For all $1 \le i \le n$ there is a sort dom_i of the tokens that may be placed at p_i. For all $1 \le j \le k$, arc_j is the product of the sorts of all variables occurring in labels of arcs adjacent to t_j.



$spec(N) =$ **WSET** `then`

`vissorts`	$dom_1,\dots,dom_n \quad arc_1,\dots,arc_n$
`defuncts`	$in_{i,j}, out_{j,i} : arc_j \to wset(dom_i)$
	for all $1 \le i \le n$ and $1 \le j \le k$
	$inout_{i,j} : arc_j \to wset(dom_i)$
`static` μ`-preds`	$guard_j : arc_j$ for all $1 \le j \le k$
`vars`	$x : arc_j$
`Horn axioms`	$inout_{i,j}(x) \equiv out_{j,i}(x) - in_{i,j}(x)$
	$\dots$

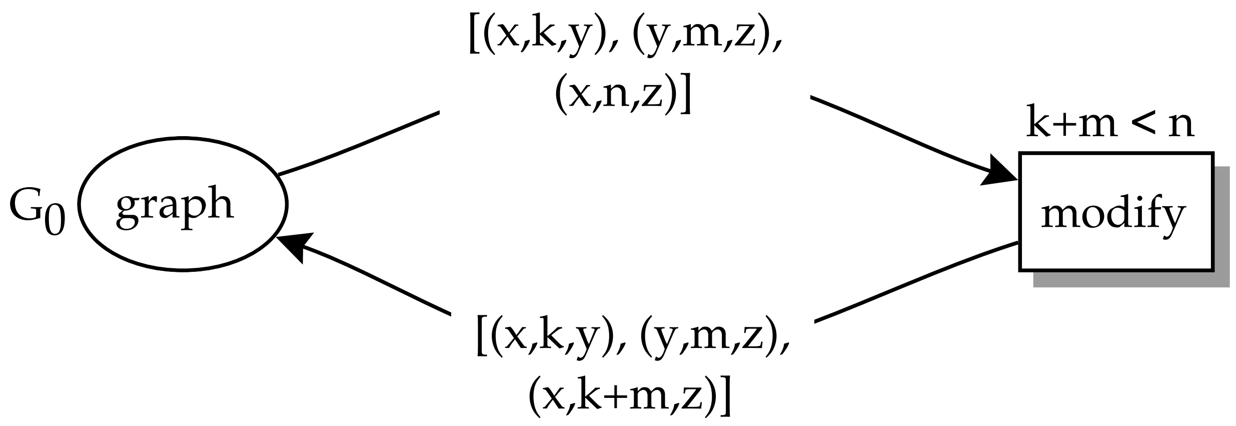


$spec(DPH) =$ **INT** `and` **WSET**$[entry \mapsto phil]$ `and`
WSET$[entry \mapsto fork]$ `then`

`vissorts`	$dom_1 = dom_3 = phil \quad dom_2 = fork$
	$dom_{take} = dom_{put} = phil$
`constructs`	$ph : int \to phil \qquad fo : int \to fork$
`defuncts`	$max :\to int \qquad LF, RF : phil \to fork$
	$in_{take,1}, out_{take,3}, out_{take,1}, in_{take,3} : phil \to phil$
	$in_{put,3}, out_{put,1}, out_{put,3}, in_{put,1} : phil \to phil$
	$out_{take,2}, in_{put,2} : phil \to fork$
`vars`	$i : int \quad x : phil$
`Horn axioms`	$LF(ph(i)) \equiv fo(i)$
	$RF(ph(i)) \equiv fo(i+1) \quad \Leftarrow \quad i < max$
	$RF(ph(max)) \equiv fo(1)$
	$in_{take,1}(x) \equiv out_{take,3}(x) \equiv in_{put,3}(x) \equiv out_{put,1}(x) \equiv [x]$
	$out_{take,1}(x) \equiv in_{take,3}(x) \equiv out_{put,3}(x) \equiv in_{put,1}(x) \equiv empty$
	$in_{take,2}(x) \equiv out_{put,2}(x) \equiv [LF(x), RF(x)]$
	$out_{take,2}(x) \equiv in_{put,2}(x) \equiv empty$
	$inout_{p,t}(x) \equiv out_{t,p}(x) - in_{p,t}(x)$
	for all $p \in \{1,2,3\}$ and $t \in \{put, take\}$

$dyn(N) = spec(N)$ `then`

`hidsorts`	$state$	
`constructs`	$succ_j : arc_j \times state \to state$	for all $1 \leq j \leq k$
`defuncts`	$p_i : state \to wset(dom_i)$	for all $1 \leq i \leq n$
`static` μ`-preds`	$enabled_j : state \times arc_j$	for all $1 \leq j \leq k$
	$enabled : state$	
	$\langle . \rangle r : state$	for all $r : state$
	$\Diamond r : state$	for all $r : state$
`transpreds`	$\_ \stackrel{\_}{\longrightarrow}_j \_ : state \times arc_j \times state$	for all $1 \leq j \leq k$
`dynamic preds`	$\_ \to \_ : state \times state$	
ν`-preds`	$disabled : state$	
	$[.]r : state$	for all $r : state$
	$\Box r : state$	for all $r : state$
`vars`	$s, s' : state \quad x : arc_j$	
`Horn axioms`	$p_i(succ_j(x, s)) \equiv p_i(s) + inout_{i,j}(x)$	
	$enabled_j(s, x) \quad \Leftarrow \quad guard_j(x) \wedge \bigwedge_{i=1}^{n} in_{i,j}(x) \subseteq p_i(s)$	
	$s \xrightarrow{x}_j succ_j(x, s) \quad \Leftarrow \quad enabled_j(s, x)$	
	$s \to s' \quad \Leftarrow \quad s \xrightarrow{x}_j s'$	
	$enabled(s) \quad \Leftarrow \quad s \to s'$	
	$\langle . \rangle r(s) \quad \Leftarrow \quad s \to s' \wedge r(s')$	
	$\Diamond r(s) \quad \Leftarrow \quad r(s)$	
	$\Diamond r(s) \quad \Leftarrow \quad s \to s' \wedge \Diamond r(s')$	
`co-Horn axioms`	$disabled(s) \quad \Rightarrow \quad (s \to s' \Rightarrow FALSE)$	
	$[.]r(s) \quad \Rightarrow \quad (s \to s' \Rightarrow r(s'))$	
	$\Box r(s) \quad \Rightarrow \quad r(s)$	
	$\Box r(s) \quad \Rightarrow \quad (s \to s' \Rightarrow \Box r(s'))$	



$$
\begin{array}{lll}
\texttt{axioms} & r(G_0) & \\
& q(s) \quad \Leftarrow & disabled(s) \wedge \\
& & \lambda(x,k,y).eq(k, mindist(G_0,x,y)) \equiv f \wedge \\
& & forall(f, graph(s)) \\
\texttt{theorem} & r(s) \quad \Rightarrow & \Diamond q(s)
\end{array}
$$

coinduction on $\Box r$

$$\frac{\varphi(s) \Rightarrow \Box r(s)}{\exists q: \ \varphi(s) \Rightarrow q(s) \quad \wedge \quad q(s) \Rightarrow r(s) \quad \wedge \quad q(s) \Rightarrow (s \rightarrow s' \Rightarrow q(s'))} \Updownarrow$$

invariant rule

$$\frac{\Box r(s)}{\begin{array}{c} \sum_{i=1}^{n} f_i(p_i(s)) \equiv \sum_{i=1}^{n} f_i(p_i(s')) \quad \Rightarrow \quad r(s') \\ \wedge \ \bigwedge_{j=1}^{k}(guard_j(x) \ \Rightarrow \ \sum_{i=1}^{n} f_i(inout_{i,j}(x)) \equiv 0) \end{array}} \Uparrow$$

if for all $1 \leq i \leq n$, $Her(spec(N))$ interprets dom as an Abelian group and $f_i : wset(dom_i) \rightarrow dom$ as a linear function

Signature morphism

Given signatures $\Sigma = (S, F, P)$ and $\Sigma' = (S', F', P')$, a *signature morphism* $\sigma : \Sigma \to \Sigma'$ consists of functions

$$\sigma_S : S \to S', \quad \sigma_F : F \to F', \quad \sigma_P : P \to P'$$

such that for all $f : s_1 \dots s_n \to s \in F$ and $r : s_1 \dots s_n \in P$,

$$\sigma_F(f) : \sigma_S(s_1) \dots \sigma_S(s_n) \to \sigma_S(s),$$
$$\sigma_P(r) : \sigma_S(s_1) \dots \sigma_S(s_n).$$

Given a Σ'-structure B, the *σ-reduct* $B|_\sigma$ is the Σ-structure that is defined by:

$$(B|_\sigma)_s = B_{\sigma(s)} \quad \text{for all } s \in S,$$
$$f^{B|_\sigma} = \sigma(f)^B \quad \text{for all } f \in F \cup P.$$

B_σ denotes the *least Σ-substructure* of $B|_\sigma$.

Relations between specifications

Let $SP = (\Sigma, AX)$ *and* $SP' = (\Sigma', AX')$ *be swinging specifications,* $\sigma : \Sigma \to \Sigma'$ *be a signature morphism and* $r : w$ *be a predicate of* Σ*. Let* $\Sigma = (S, F, P)$ *and* A *be a* Σ'*-structure.*

SP' is *r-monotone w.r.t. (SP, σ)* if for all $t \in T_{\Sigma,w}$,

$$Her(SP) \models r(t) \quad \Rightarrow \quad Her(SP') \models \sigma(r(t)).$$

SP' is *r-consistent w.r.t. (SP, σ)* if for all $t \in T_{\Sigma,w}$,

$$Her(SP') \models \sigma(r(t)) \quad \Rightarrow \quad Her(SP) \models r(t).$$

SP' is *monotone* resp. *consistent w.r.t. (SP, σ)* if for all predicates $r \in \Sigma$, SP' is r-monotone resp. r-consistent w.r.t. (SP, σ).

SP' *refines SP along σ* if $Her(SP')_\sigma$ is an SP-model and SP' is consistent w.r.t. (SP, σ).
σ is called the *representation morphism* of the refinement.

Refinement: Model change via signature morphisms

Let A be the Herbrand model of an *abstract* specification $SP = (\Sigma, AX)$ and B be the Herbrand model of a *concrete* specification $SP' = (\Sigma', AX')$. *SP' refines SP along σ* if

(1) the *refinement model* B_σ satisfies AX,

(2) SP' is *consistent w.r.t.* SP, i.e., negation-free formulas that are valid in B_σ are also valid in A.

If SP *has complement predicates* and σ *preserves negation*, then (2) follows from (1).

☞ Refinements make a *loose semantics* obsolete.

☞ Structural equalities may be refined to behavioral ones. $\rightsquigarrow$ Visible sorts are mapped to hidden ones.

☞ Non-composite states may be refined to composite ones.

☞ Global operations may be refined to message passing primitives by mapping the operations' — *static* — I/O relations to — *dynamic* — transition relations.

☞ Atomic states may be refined to composite ones.

Polymorphism and parameterization

A (formal) *parameter specification* $PAR = (P\Sigma, PAX)$ consists of a signature $P\Sigma$ and a set PAX of first-order $P\Sigma$-formulas.

Given a swinging specification $SP = (\Sigma, AX)$ with $P\Sigma \subseteq \Sigma$, $PSP = (PAR, SP)$ is a *parameterized specification*. Sorts of SP may be terms built up of sort constructors and sorts of $P\Sigma$.

ENTRY

`sorts`	$entry$
`preds`	$\_ \not\equiv \_ : entry \times entry$
`vars`	$x, y : entry$
`axioms`	$x \not\equiv y \Leftrightarrow \neg(x \equiv y)$

LIST $=$ **ENTRY** `and` **NAT** `then`

`vissorts`	$list = list(entry)$
`constructs`	$nil : \rightarrow list$
	$\_ :: \_ : entry \times list \rightarrow list$
`defuncts`	$nth : list \times nat \rightarrow 1 + entry$
`vars`	$n : nat \quad x : entry \quad L : list$
`Horn axioms`	$nth(nil, n) \equiv ()$
	$nth(x :: L, 0) \equiv (x)$
	$nth(x :: L, n+1) \equiv nth(L, n)$

Let $\sigma : P\Sigma \rightarrow \Sigma'$ be a signature morphism with finite domain, say $dom(\sigma) = \{s_1, \ldots, s_n\}$. The *actualization $SP[\sigma]$ of SP along σ*, written as

$$SP[s_1 \mapsto \sigma(s_1), \ldots, s_n \mapsto \sigma(s_n)],$$

is the specification obtained from SP by replacing, for all $s \in dom(\sigma)$, all occurrences of s in SP by $\sigma(s)$.

$SP[\sigma]$ is *correct w.r.t.* a specification $SP' = (\Sigma', AX')$ if $\sigma(P\Sigma) \subseteq \Sigma'$ and $Her(SP') \models \sigma(PAX)$.

The *Herbrand SP-model, $Her(SP)$*, is the class of Herbrand models of actualizations of SP. $Her(SP)$ *has property P* if all actualizations of SP have property P.

Consistency criteria

Let $SP = (\Sigma, AX)$ *and* $SP' = (\Sigma', AX')$ *be swinging specifications and* $\sigma : \Sigma \to \Sigma'$ *be a signature morphism.*

- *Let* $Her(SP')_\sigma$ *be an* SP*-model. For all* $r \in \mu P$, SP' *is* r*-monotone w.r.t.* (SP, σ), *and for all* $r \in \nu P$, SP' *is* r*-consistent w.r.t.* (SP, σ).
- *Let* σ *be an inclusion and* R *be a subset of* νP *resp.* μP *such that the axioms of* SP *for* R *coincide with the axioms of* SP' *for* R. *Then for all* $r \in R$, SP' *is* r*-monotone resp.* r*-consistent w.r.t.* SP.
- SP' *is consistent w.r.t.* (SP, σ) *if* SP' *is complete and confluent, all axioms of* $SP' \setminus \sigma(SP)$ *are axioms for symbols of* $SP' \setminus \sigma(SP)$ *and* $\sigma(NF_\Sigma) = NF_{\Sigma'}$.
- SP' *is consistent w.r.t.* (SP, σ) *if* $Her(SP')_\sigma$ *is an* SP*-model, for all* $r \in P$, *the complement* $\overline{r}$ *of* r *w.r.t.* $Her(SP)$ *is in* P *and* $\sigma(\overline{r})$ *is the complement of* $\sigma(r)$ *w.r.t.* $Her(SP')$, *and for all* $r \in \mu P$, $\overline{r} \in \mu P$ *or* SP' *is* r*-consistent w.r.t.* (SP, σ).

$SP = \mathbf{STACK} = \mathbf{ENTRY}$ `then`

`vissorts`	$stack$
`constructs`	$() : \to 1$
	$empty : \to stack$
	$push : entry \times stack \to stack$
`defuncts`	$pop : stack \to stack$
	$top : stack \to 1 + entry$
`static` μ-`preds`	$\not\equiv : stack \times stack$
`vars`	$x : entry \quad s : stack$
`Horn axioms`	$top(empty) \equiv ()$
	$pop(empty) \equiv empty$
	$top(push(x,s)) \equiv (x)$
	$pop(push(x,s)) \equiv s$
	$empty \not\equiv push(x,s)$
	$push(x,s) \not\equiv empty$
	$push(x,s) \not\equiv push(y,s') \quad \Leftarrow \quad x \not\equiv y$
	$push(x,s) \not\equiv push(y,s') \quad \Leftarrow \quad s \not\equiv s'$

$SP' = \mathbf{ENTRY}$ `and` $\mathbf{NAT}$ `then`

`vissorts`	$nat + entry$
`hidsorts`	$map \quad stack$
`constructs`	$\kappa_1 : nat \to nat + entry$
	$\kappa_2 : entry \to nat + entry$
	$new : \to map$
	$upd : nat \times entry \times map \to map$
	$mkStack : map \times nat \to stack$
`destructs`	$get : map \times nat \to nat + entry$
	$top : stack \to nat + entry$
	$pop : stack \to stack$
`defuncts`	$pred : nat \to nat$
	$empty : \to stack$
	$push : entry \times stack \to stack$
`static` μ-`preds`	$\not\sim : stack \times stack$

$$\not\equiv : (nat + entry) \times (nat + entry)$$

`vars` $\quad i, j : nat \quad x, y : entry \quad f : map \quad s, s' : stack$

`Horn axioms`

$$pred(0) \equiv 0$$
$$pred(i+1) \equiv i$$
$$get(new, i) \equiv \kappa_1(i)$$
$$get(upd(i, x, f), i) \equiv \kappa_2(x)$$
$$get(upd(i, x, f), j) \equiv get(f, j) \quad \Leftarrow \quad i \not\equiv j$$
$$empty \equiv mkStack(new, 0)$$
$$push(x, mkStack(f, i)) \equiv mkStack(upd(i+1, x, f), i+1)$$
$$top(mkStack(f, i)) \equiv get(f, i)$$
$$pop(mkStack(f, i)) \equiv mkStack(f, pred(i))$$
$$s \not\sim s' \quad \Leftarrow \quad top(s) \not\equiv top(s')$$
$$s \not\sim s' \quad \Leftarrow \quad pop(s) \not\sim pop(s')$$
$$\kappa_1(i) \not\equiv \kappa_2(x)$$
$$\kappa_2(x) \not\equiv \kappa_1(i)$$
$$\kappa_1(i) \not\equiv \kappa_1(j) \quad \Leftarrow \quad i \not\equiv j$$
$$\kappa_2(x) \not\equiv \kappa_2(y) \quad \Leftarrow \quad x \not\equiv y$$

$$
\begin{aligned}
rep(\equiv : stack \times stack) &= \sim : stack \times stack \\
rep(\not\equiv : stack \times stack) &= \not\sim : stack \times stack \\
rep(1) &= nat \\
rep(() \mapsto 1) &= 0 \mapsto nat, \\
rep(s) &= s \quad \text{for all other symbols of } SP
\end{aligned}
$$

<u>*SP' refines SP along σ:*</u>

$$
\begin{aligned}
&top(empty) \equiv \kappa_1(0) && (1)\\
&pop(empty) \sim empty && (2)\\
&top(push(x,s)) \equiv \kappa_2(x) && (3)\\
&pop(push(x,s)) \sim s && (4)\\
&empty \not\sim push(x,s) && (5)\\
&push(x,s) \not\sim push(y,s') \quad \Leftarrow \quad x \not\equiv y && (6)\\
&push(x,s) \not\sim push(y,s') \quad \Leftarrow \quad s \not\sim s' && (7)
\end{aligned}
$$

are inductive theorems of SP'. For all predicates r of SP, the complement $\overline{r}$ of r w.r.t. $Her(SP)$ is in SP, $\sigma(\overline{r})$ is the complement of $\sigma(r)$ w.r.t. $Her(SP')$. The complements of μ-predicates of SP are μ-predicates.

SP is coinductive. SP' is coinductive if

$$s \sim t \quad \Rightarrow \quad push(x,s) \sim push(x,t) \qquad (8)$$

is an inductive theorem of SP'. A proof of (4) employs the lemma:

$$mkStack(upd(i,x,f),j) \sim mkStack(f,j) \;\Leftarrow\; i > j \qquad (9)$$

that characterizes behavioral stack equivalence in terms of the refinement constructor $mkStack$.

B is **monotone wrt** A if

- for all ground static μ-atoms p,

$$A \models p \quad \Rightarrow \quad B \models p,$$

- for all ground dynamic atoms $\delta(t, u)$,

$$A \models \delta(t, u) \quad \Rightarrow \quad B \models \delta(t, v)$$

  for some $v \in T_\Sigma$ with $u^A = v^A$,

- for all ground ν-atoms p,

$$B \models p \quad \Rightarrow \quad A \models p.$$

(Sel, Sim)-unfolding

$$\frac{G}{\exists Y((G_1 \wedge \bigwedge_{x \in dom(\tau_1)} x \equiv x\tau_1) \vee \cdots \vee (G_n \wedge \bigwedge_{x \in dom(\tau_n)} x \equiv x\tau_n))} \Updownarrow$$

if Sel is a uniform redex selector, Sim is a $>$-reductive simplifier,

$succ_{Sim}^{Sel}\langle G, id\rangle = \{\langle G_1, \tau_1\rangle, \ldots, \langle G_n, \tau_n\rangle\}$,

$Y = \uplus_{i=1}^{n} var(dom(\tau_i)\tau_i)$ and $Y \cap (var(G) \cup dom(\tau_i)) = \emptyset$

Horn clause completion SP_∞

Aim: turning liveness into safety predicates

For each $r : w \in live(SP)$, the approximating predicate $r' : nat \times w$ is added to SP and each co-Horn axiom

$$r(x) \Rightarrow (H \Rightarrow (\exists X_1 G_1 \vee \cdots \vee \exists X_n G_n))$$

is replaced by the generalized Horn axioms

$$\begin{array}{rcl} r(x) & \Leftarrow & \forall i : r'(i,x) \\ r'(0,x) & & \\ r'(i+1,x) & \Leftarrow & \forall Y_1(H \Rightarrow \exists X_1 G_1[q'(i,u)/q(u) \mid q \in live(SP)] \\ & \ldots & \\ r'(i+1,x) & \Leftarrow & \forall Y_n(H \Rightarrow \exists X_n G_n[q'(i,u)/q(u) \mid q \in live(SP)] \end{array}$$

where $Y_i = var(G_i) \setminus X_i \setminus \{x\}$.

A proof by induction on i shows that for all $i \in \mathbb{N}$, live levels k of SP, $r : w \in \Sigma_k \setminus \Sigma_{k-1}$ and $t \in T_{\Sigma,w}$,

$$\Psi_k^i(\top) \models r(t) \quad \Longleftrightarrow \quad Ini(SP_\infty) \models r'(i,t).$$

Hence for all ground atoms $r(t)$ over Σ,

$$Sem(SP) \models r(t) \quad \Longleftrightarrow \quad Ini(SP_\infty) \models r(t)$$

and thus

$$Sem(SP) \;\cong\; Ini(SP_\infty)$$

if SP is **downward continuous**, i.e. for all live levels k of SP, Ψ_k is downward continuous.

iff-Completions

Let r be a safety predicate and f be a defined function,

$$AX_r = \{r(t_i) \Leftarrow H_i \mid 1 \leq i \leq n\},$$
$$AX_f = \{f(t_i) \equiv u_i \Leftarrow H_i \mid 1 \leq i \leq n\}$$

be the sets of axioms for r resp. f,
$Y_i = var(r(t_i) \Leftarrow H_i)$ resp. $Y_i = var(f(t_i) \equiv u_i \Leftarrow H_i)$
and x, y be (tuples of) variables that do not occur in Y_i.

$$r(x) \iff \bigvee_{i=1}^{n} \exists Y_i(x \equiv t_i \wedge H_i),$$
$$f(x) \equiv y \iff \bigvee_{i=1}^{n} \exists Y_i(x \equiv t_i \wedge y \equiv u_i \wedge H_i)$$

are the **iff-completions** of r and f, respectively.

Let r be a liveness predicate and

$$AX_r = \{r(x) \Rightarrow (H_i \Rightarrow \exists X_i GS_i) \mid 1 \leq i \leq n\}$$

be the set of axioms for r and $Y_i = var(GS_i) \setminus X_i \setminus \{x\}$.

$$r(x) \iff \bigwedge_{i=1}^{n} \forall Y_i(H_i \Rightarrow \exists X_i GS_i)$$

is the **iff-completion** of r.

Iff-completions are final theorems of SP.

$$\begin{aligned} sorted(L) \iff & L \equiv nil \ \vee \ \exists x : (L \equiv [x]) \vee \\ & \exists x, y, L' : (L \equiv x :: y :: L' \wedge x \leq y \wedge sorted(y :: L')) \end{aligned}$$

is the iff-completion of $sorted : list$, specified by:

$$\begin{aligned} sorted(nil) & \\ sorted([x]) & \\ sorted(x :: y :: L) & \Leftarrow \ x \leq y \wedge sorted(y :: L). \end{aligned}$$

$$s \sim s' \iff \forall x, t : (s \xrightarrow{x} t \ \Rightarrow \ \exists t' : (s' \xrightarrow{x} t' \wedge t \sim t')) \ \wedge \ s' \sim s$$

is the iff-completion of $\sim$: $stream \times stream$, specified by:

$$s \sim s' \ \Rightarrow \ (s \xrightarrow{x} t \Rightarrow \exists t' : (s' \xrightarrow{x} t' \wedge t \sim t')).$$

Complements

Given predicates $r : w$ and $\overline{r} : w$ of SP, $\overline{r}$ is the **SP-complement** of r if $\overline{r}^{Fin(SP)}$ is the complement of $r^{Fin(SP)}$ i.e.

$$\overline{r}^{Fin(SP)} = Fin(SP)_w \setminus r^{Fin(SP)}.$$

Theorem Given a safety (resp. liveness) predicate r and a liveness (resp. safety) predicate $\overline{r}$ such that $r(x) \Leftrightarrow C_r$ and $\overline{r}(x) \Leftrightarrow C_{\overline{r}}$ are the iff-completions of r and $\overline{r}$, respectively, and

$$C_{\overline{r}} \iff \neg C_r[\overline{p}(t)/\neg p(t) \mid p \in PR].$$

If SP is downward continuous, then $\overline{r}$ is the SP-complement of r.

By negating the iff-completion of *sorted* we obtain:

$$\begin{aligned} unsorted(L) \iff\ & L \not\equiv nil \ \wedge\ \forall x : (L \not\equiv [x]) \ \wedge \\ & \forall x, y, L' : (L \not\equiv x :: y :: L' \vee x \not\leq y \vee unsorted(y :: L')). \end{aligned}$$

Hence the following co-Horn axioms specify the complement of *sorted*:

$$\begin{aligned} & unsorted(L) \ \Rightarrow\ (L \equiv nil \ \Rightarrow\ FALSE) \\ & unsorted(L) \ \Rightarrow\ (L \equiv [x] \ \Rightarrow\ FALSE) \\ & unsorted(L) \ \Rightarrow\ ((L \equiv x :: y :: L' \wedge x \leq y) \ \Rightarrow\ unsorted(y :: L')). \end{aligned}$$

Given a goal G, **safe(G)** and **live(G)** denote the sets of atoms $r(t)$ of G such that r is a safety resp. liveness predicate.

Let $R \subseteq live(SP)$ and HC be a set of Horn clauses $r(t) \Leftarrow H$ such that $r \in R$, $live(H) \subseteq R$ and $Sem(SP) \models HC$.

HC is **>-reductive** if for all $r \in R$, $(r(t) \Leftarrow H) \in HC$ and $\sigma \in T_\Sigma^X$,

$$fresh(r(t) \Leftarrow H)\sigma \subseteq NF_\Sigma \ \wedge \ safe(H)\sigma \vdash_{SP} \emptyset \quad \Rightarrow \quad r(t\sigma) > H\sigma$$

Theorem Suppose that SP is terminating and confluent, SP is downward continuous and R and HC are as above.

Construct HSP from SP by replacing the co-Horn axioms for R by HC.

If HC is >-reductive, then $Fin(HSP)$ and $Fin(SP)$ are isomorphic.

The co-Horn axioms:

$$unsorted(L) \ \Rightarrow \ (L \equiv nil \ \Rightarrow \ FALSE)$$

$$unsorted(L) \ \Rightarrow \ (L \equiv [x] \ \Rightarrow \ FALSE)$$

$$unsorted(L) \ \Rightarrow \ ((L \equiv x :: y :: L' \wedge x \leq y) \ \Rightarrow \ unsorted(y :: L'))$$

become Horn axioms:

$$unsorted(x :: y :: L) \Leftarrow x > y$$
$$unsorted(x :: y :: L) \Leftarrow unsorted(y :: L).$$

Consistency criteria

Let v be a term or an atom with a single occurrence of a variable x,

$$C \;=\; t\{\equiv u\} \Leftarrow H \quad \text{and} \quad C' \;=\; t' \equiv u' \Leftarrow H'$$

be Horn axioms of SP and σ, τ be minimal substitutions such that

$$t\sigma \;=\; v[t'\tau/x]$$

and the position of x in v agrees with the position of a function symbol in t. The Horn clause

$$CC \;=\; v[u'\tau/x]\{\equiv u\sigma\} \;\Leftarrow\; H\sigma \wedge H'\tau$$

is the **SP-critical clause induced by C on C' at $t\sigma$**.

SP is **weakly orthogonal** if for each SP-critical clause $t\{\equiv t'\} \Leftarrow H$, $t = t'$ or $Ini(SP) \models \neg H$.

Given a binary relation $>$ on terms and goals, a Horn clause $p \Leftarrow H$ is **subjoinable below** a term or atom t if

$$H\sigma \vdash_{SP} \emptyset \;\;\Rightarrow\;\; p\sigma \vdash_{SP} \emptyset$$

for all $\sigma \in T_\Sigma^X$ such that all SP-convergent ground goals $G < t\sigma$ are strongly SP-convergent.

Superposition Theorem A terminating specification is confluent iff for all t, each SP-critical clause induced at t is subjoinable below t.

Consistency criteria II

(1) **(proof by consistency)** Let $\Sigma = \Sigma'$.
$Ini(SP)$ is consistent wrt $Ini(SP')$
iff $SP \vdash_{cut} p$ implies $SP' \vdash_{cut} p$ for all ground atoms p over SP'
iff $AX \setminus AX'$ consists of inductive theorems of SP'.

(2) $Ini(SP)$ is consistent wrt $Ini(SP')$ if

- $s \in \Sigma \setminus \Sigma'$ for all function symbols $f : w \to s \in \Sigma \setminus \Sigma'$,
- $r \in \Sigma \setminus \Sigma'$ for all $(r(t) \Leftarrow H) \in AX \setminus AX'$.

(3) $Ini(SP)$ is consistent wrt $Ini(SP')$ if

- SP is functional,
- $f \in \Sigma \setminus \Sigma'$ for all $(f(t)\{\equiv u\} \Leftarrow H) \in AX \setminus AX'$,
- $s \in \Sigma \setminus \Sigma'$ for all constructors $c : w \to s \in \Sigma \setminus \Sigma'$ **or** SP' is complete and all axioms of SP' are deterministic and see MST.

A specification of weighted sets

$WS(s_1, \ldots, s_n) = \mathrm{NAT} + \mathrm{INT} + \mathrm{BOOL} +$

`vissorts`	$s_1 \ldots s_n$	
	$sum = s_1 + \cdots + s_n$	
`hidsorts`	$wset = wset(sum)$	
`consts`	$\_ : s_i \rightarrow sum$	for all $\leq i \leq n$
	$empty : \rightarrow wset$	
	$[\_] : sum \rightarrow wset$	
	$\_ + \_ : wset \times wset \rightarrow wset$	
	$-\_ : wset \rightarrow wset$	
	$map : (sum \rightarrow sum) \rightarrow wset \rightarrow wset$	
	$filter : (sum \rightarrow bool) \rightarrow wset \rightarrow wset$	
`defuns`	$weight : sum \times wset \rightarrow int$	
	$\_ * \_ : nat \times sum \rightarrow wset$	
	$\_ - \_ : wset \times wset \rightarrow wset$	
	$\lvert\_\rvert : wset \rightarrow int$	
`preds`	$\_ \in \_ : sum \times wset$	
	$\_ \subseteq \_ : wset \times wset$	
	$\_ \sim \_ : wset \times wset$	
`vars`	$V, W : wset \quad x, y : sum \quad n : nat$	
`Horn axioms`	$weight(x, empty) \equiv 0$	
	$weight(x, [x]) \equiv 1$	
	$weight(x, [y]) \equiv 0 \quad \Leftarrow \quad x \not\equiv y$	
	$weight(x, V + W) \equiv weight(x, V) + weight(x, W)$	
	$weight(x, -W) \equiv -weight(x, W)$	
	$weight(x, map(f)(W)) \equiv \sum_{y:f(y)=x} weight(y, W)$	
	$weight(x, filter(g)(W)) \equiv weight(x, W)$ $\quad \Leftarrow \quad g(x) \equiv true$	
	$weight(x, filter(g)(W)) \equiv 0 \quad \Leftarrow \quad g(x) \equiv false$	
	$0 * x \equiv empty$	
	$(n + 1) * x \equiv [x] + (n * x)$	
	$V - W \equiv V + (-W)$	

$$
\begin{array}{ll}
 & |W| \equiv \sum_x weight(x, W) \\
 & x \in W \quad \Leftarrow \quad weight(x, W) > 0 \\
\texttt{co-Horn axioms} & V \subseteq W \quad \Rightarrow \quad weight(x, V) \leq weight(x, W) \\
 & V \sim W \quad \Rightarrow \quad weight(x, V) \equiv weight(x, W)
\end{array}
$$

COMMAND = BOOL + INT +

`vissorts`	$var \;\; val \;\; exp$
`hidsorts`	$com \;\; state \;\; conf$
`consts`	$\_ := \_ : var \times exp \to com$
	$\_? : exp \to com$
	$\_;\_ : com \times com \to com$
	$\_ + \_ : com \times com \to com$
	$\_^{*} : com \to com$
	$new : \to state$
	$fail : \to state$
	$upd : var \times val \times state \to state$
	$\_;\_ : state \times com \to state$
	$\_ : var \to exp$
	$\_ : val \to exp$
	$\_ : int \to val$
	$\_ : bool \to val$
`defuns`	$skip : \to com$
	$cond : exp \times com \times com \to com$
	$while : exp \times com \to com$
	$until : com \times exp \to com$
	$get : map \times var \to val$
	$eval : map \times exp \to val$
`preds`	$\_ \longrightarrow \_ : conf \times conf$
	$\_ \stackrel{*}{\longrightarrow} \_ : conf \times state$
	$success, failure : state$
	$\_ \sim \_ : state \times state$
	$\_ \sim \_ : conf \times conf$
`vars`	$c, c', d, d' : com \quad x, y : var \quad i : val \quad e : exp$
	$\kappa, \kappa', \delta, \delta' : conf \quad s, s', t : state$
`Horn axioms`	$\kappa \stackrel{*}{\longrightarrow} s \;\Leftarrow\; \kappa \longrightarrow \delta \wedge \delta \stackrel{*}{\longrightarrow} s$
	$s; x := e \stackrel{*}{\longrightarrow} upd(x, eval(s, e), s)$
	$s; e? \stackrel{*}{\longrightarrow} s \;\Leftarrow\; eval(s, e) \equiv true$
	$s; e? \stackrel{*}{\longrightarrow} fail \;\Leftarrow\; eval(s, e) \equiv false$

$$s;(c;c') \longrightarrow t;c' \;\Leftarrow\; s;c \xrightarrow{*} t \wedge success(t)$$
$$s;(c;c') \longrightarrow t \;\Leftarrow\; s;c \xrightarrow{*} t \wedge failure(t)$$
$$s;(c+c') \longrightarrow \kappa \;\Leftarrow\; s;c \longrightarrow \kappa$$
$$s;(c+c') \longrightarrow \kappa \;\Leftarrow\; s;c' \longrightarrow \kappa$$
$$s;(c+c') \xrightarrow{*} t \;\Leftarrow\; s;c \xrightarrow{*} t$$
$$s;(c+c') \xrightarrow{*} t \;\Leftarrow\; s;c' \xrightarrow{*} t$$
$$s;c^* \longrightarrow t;c^* \;\Leftarrow\; s;c \xrightarrow{*} t \wedge success(t)$$
$$s;c^* \xrightarrow{*} s \;\Leftarrow\; s;c \xrightarrow{*} t \wedge failure(t)$$
$$skip \equiv true?$$
$$cond(e,c,c') \equiv (e?.c) + (\neg e?.c')$$
$$while(e,c) \equiv (e?;c)^*;\neg e?$$
$$until(c,e) \equiv c;while(\neg e,c)$$
$$success(new)$$
$$success(upd(x,i,s)) \;\Leftarrow\; success(s)$$
$$failure(fail)$$
$$get(new,x) \equiv \bot$$
$$get(fail,x) \equiv \bot$$
$$get(upd(x,i,s),x) \equiv i$$
$$get(upd(x,i,s),y) \equiv get(s,y) \;\Leftarrow\; x \not\equiv y$$

axioms for $eval$

`co-Horn axioms`

$$\kappa \sim \kappa' \;\Rightarrow\; (\kappa \longrightarrow \delta \;\Rightarrow\; \exists \delta'(\kappa' \longrightarrow \delta' \wedge \delta \sim \delta'))$$
$$\kappa \sim \kappa' \;\Rightarrow\; (\kappa \xrightarrow{*} s \;\Rightarrow\; \exists s'(\kappa' \xrightarrow{*} s' \wedge s \sim s'))$$
$$s \sim s' \;\Rightarrow\; get(s,x) \equiv get(s',x)$$
$$s \sim s' \;\Rightarrow\; (success(s) \;\Rightarrow\; success(s'))$$
$$s \sim s' \;\Rightarrow\; (failure(s) \;\Rightarrow\; failure(s'))$$

PROCESS = BOOL + INT +

`vissorts`	$port \ \ act \ \ var \ \ exp$
`hidsorts`	$proc$
`consts`	$\_?\!:\!port \rightarrow act$
	$\_!\!:\!port \rightarrow act$
	$\tau :\rightarrow act$
	$stop :\rightarrow proc$
	$\_.\_\!:\!act \times proc \rightarrow proc$
	$\_ + \_\!:\!proc \times proc \rightarrow proc$
	$\_\vert\_\!:\!proc \times proc \rightarrow proc$
	$\_ \setminus \_\!:\!proc \times port \rightarrow proc$
	$map\!:\!(act \rightarrow act) \times proc \rightarrow proc$
	$\_?\_\!:\!port \times exp \rightarrow act$
	$\_!\_\!:\!port \times int \rightarrow act$
	$\_\!:\!var \rightarrow exp$
	$\_\!:\!int \rightarrow exp$
	$\_\!:\!bool \rightarrow exp$
`defuns`	$\overline{\_}\!:\!act \rightarrow act$
	$\_[\_/\_]\!:\!proc \times int \times var \rightarrow proc$
	$\_[\_/\_]\!:\!exp \times int \times var \rightarrow exp$
`preds`	$\_ \stackrel{\_}{\longrightarrow} \_ : proc \times act \times proc$
	$empty : proc$
	$\_ \sim \_ : proc \times proc$
`vars`	$a, b : act \ \ p, p', q, q' : proc \ \ f\!:\!act \rightarrow act \ \ c : port$
	$x, y : var \ \ i, j : int \ \ e, e' : exp$
`Horn axioms`	$\overline{c?} \equiv c!$
	$\overline{c!} \equiv c?$
	$\overline{\tau} \equiv \tau$
	$\overline{c?e} \equiv c!e$
	$\overline{c!e} \equiv c?e$
	$a.p \stackrel{a}{\longrightarrow} p$
	$a.\tau.p \stackrel{a}{\longrightarrow} p$
	$\tau.p \stackrel{a}{\longrightarrow} q \ \Leftarrow \ p \stackrel{a}{\longrightarrow} q$

$$p \xrightarrow{a} \tau.q \;\Leftarrow\; p \xrightarrow{a} q$$
$$p + p' \xrightarrow{a} q \;\Leftarrow\; p \xrightarrow{a} q$$
$$p + p' \xrightarrow{a} q \;\Leftarrow\; p' \xrightarrow{a} q$$
$$p|p' \xrightarrow{a} q|p' \;\Leftarrow\; p \xrightarrow{a} q$$
$$p|p' \xrightarrow{a} p|q' \;\Leftarrow\; p' \xrightarrow{a} q'$$
$$p|p' \xrightarrow{\tau} q|q' \;\Leftarrow\; p \xrightarrow{a} q \wedge p' \xrightarrow{b} q' \wedge a \equiv \overline{b} \wedge a \not\equiv \tau$$
$$p \setminus b \xrightarrow{a} q \setminus b \;\Leftarrow\; p \xrightarrow{a} q \wedge a \not\equiv b \wedge a \not\equiv \overline{b}$$
$$map(f,p) \xrightarrow{f(a)} map(f,q) \;\Leftarrow\; p \xrightarrow{a} q$$
$$c?x.p \xrightarrow{c?i} p[i/x]$$
$$stop[i/x] \equiv stop$$
$$(c!e.p)[i/x] \equiv c!e[i/x].p[i/x]$$
$$(c?x.p)[i/x] \equiv c?x.p$$
$$(c?x.p)[i/y] \equiv c?x.p[i/y] \;\Leftarrow\; x \not\equiv y$$
$$(p + p')[i/x] \equiv p[i/x] + p'[i/x]$$
$$x[i/x] \equiv i$$
$$x[i/y] \equiv y \;\Leftarrow\; x \not\equiv y$$
$$(e + e')[i/x] \equiv e[i/x] + e'[i/x]$$

further axioms for the substitution operators

$$empty(stop)$$
$$empty(p + p') \;\Leftarrow\; empty(p) \wedge empty(p')$$
$$empty(p[i/x]) \;\Leftarrow\; empty(p)$$

further axioms for *empty*

`co-Horn axioms`

$$p \sim p' \;\Rightarrow\; (p \xrightarrow{a} q \;\Rightarrow\; \exists q'(p' \xrightarrow{a} q' \wedge q \sim q'))$$
$$p \sim p' \;\Rightarrow\; (empty(p) \;\Rightarrow\; empty(p'))$$
$$p \sim p' \;\Rightarrow\; p[i/x] \sim p'[i/x]$$

τ-laws

$$a.\tau.p \;\equiv\; a.p \tag{1}$$

$$p + \tau.p \;\equiv\; \tau.p \tag{2}$$

$$a.(p + \tau.q) + a.q \;\equiv\; a.(p + \tau.q) \tag{3}$$

expansion law

$$p \equiv \sum_{i=1}^{n} a_i p_i \;\wedge\; q \equiv \sum_{i=1}^{n} b_i q_i \quad\Rightarrow\quad (p|q) \setminus c \;\equiv\; \begin{cases} \sum_{i=1}^{n} a_i(p_i|q) \setminus c+ \\ \sum_{i=1}^{n} b_i(p|q_i) \setminus c \;+ \\ \sum_{i<j:p_i=\overline{q_j},p_i\neq\tau} \tau.(p_i|q_j) \setminus c \end{cases} \tag{4}$$

unique-solution law.

$$\begin{aligned} & P_1 \equiv \textstyle\sum_{j=1}^{n_1} a_{1j}p_{1j} \;\wedge\; \ldots \;\wedge\; P_k \equiv \sum_{j=1}^{n_k} a_{kj}p_{kj} \;\wedge \\ & Q_1 \equiv \textstyle\sum_{j=1}^{n_1} a_{1j}p_{1j}[Q_1/P_1, \ldots, Q_k/P_k] \;\wedge\; \ldots \;\wedge \\ & Q_k \equiv \textstyle\sum_{j=1}^{n_k} a_{kj}p_{kj}[Q_1/P_1, \ldots, Q_k/P_k] \\ \Rightarrow\; & P_1 \equiv Q_1 \;\wedge\; \ldots \;\wedge\; P_k \equiv Q_k \end{aligned} \tag{5}$$

BUFFERS = PROCESS +

`consts`	$buf_1, buf_2, buf_3 :\rightarrow proc$
	$\alpha, \beta, \gamma :\rightarrow port$
`Horn axioms`	$buf_1 \xrightarrow{\alpha?} \gamma!.buf_1$
	$buf_2 \xrightarrow{\gamma?} \beta!.buf_2$
	$buf_3 \xrightarrow{\alpha?} \beta!.buf_3$
	$buf_3 \xrightarrow{\beta!} \alpha?.buf_3$

$$(buf_1|buf_2) \setminus \gamma? \;\equiv\; \alpha?.buf_3 \tag{6}$$

Partial-recursive functions

Aim: extend SP by a function symbol $f : w \to s$ with "recursive" axioms AX_f and interpreting f as the least function satisfying AX_f with respect to the partial order of increasing definedness:

$$g \;\leq\; h \quad \Longleftrightarrow_{def} \quad Sem(SP) \models \forall x\ (g(x) \equiv \bot \vee g(x) \equiv h(x)).$$

Extend SP by the sort nat, constructors $0 : \to nat$ and $\_ + 1 : nat \to nat$, the defined function $f' : nat \times w \to s$ and axioms

$$f'(0, x) \equiv \bot$$

$$f'(n+1, t) \equiv u[f'(n, v)/f(v)] \;\Leftarrow\; H[f'(n, v)/f(v)]$$

for all $(f(t) \equiv u \Leftarrow H) \in AX_f$.

Weak syntactic conditions on AX_f ensure that SP remains functional. We specify the graph $r_f : w$ of f by the axioms

$$r_f(x, y) \;\Leftarrow\; f'(n, x) \equiv y \wedge Def(y)$$

$$r_f(x, \bot) \;\Leftarrow\; \forall n : f'(n, x) \equiv \bot.$$

r_f can be turned into a function if $Ini(SP)$ and thus $\mu\Phi_4$ satisfy

$$\forall x \exists y : r_f(x, y) \tag{1}$$

$$\forall x, y, z : ((r_f(x, y) \wedge r_f(x, z)) \;\Rightarrow\; x \equiv y). \tag{2}$$

(1) follows from the completeness of SP. (2) reduces to:

$$\forall m, n, x : ((Def(f'(m, x)) \wedge Def(f'(n, x))) \;\Rightarrow\; f'(m, x) \equiv f'(n, x)). \tag{3}$$

Proof of (3)

Assume $SP \vdash_{cut} Def(f'(m,t)) \wedge Def(f'(n,t))$.

$m = n \quad \Rightarrow \quad f'(m,t) \equiv_{SP} f'(n,t)$.

$m \neq n \quad \Rightarrow$

$$\left.\begin{array}{c} f'(m,t) \equiv_{SP} u(v) \ \text{ and } \ f'(n,t) \equiv_{SP} u(\bot) \\ \text{or} \\ f'(m,t) \equiv_{SP} u(\bot) \ \text{ and } \ f'(n,t) \equiv_{SP} u(v) \end{array}\right\} \text{for some terms } u, v.$$

Hence $SP \vdash_{cut} Def(u(\bot))$. Suppose that $Ini(SP)$ satisfies

$$\forall x \ (Def(u(\bot)) \ \Rightarrow \ u(x) \equiv u(\bot)). \tag{4}$$

Then w.l.o.g.

$$f'(m,t) \equiv_{SP} u(v) \equiv_{SP} u(\bot) \equiv_{SP} f'(n,t).$$

(4) says that the functions u consists of are **continuous** in the sense of CPO theory: for all $1 \leq i \leq n$,

$$\begin{array}{l} Def(g(x_1,\dots,x_{i-1},\bot,x_{i+1},\dots,x_n)) \\ \quad \Rightarrow \ g(x_1,\dots,x_n) \equiv g(x_1,\dots,x_{i-1},\bot,x_{i+1},\dots,x_n). \end{array}$$

$f : w \to s$ is added to Σ and $Sem(SP)$ is factored through the least Σ-congruence $\approx_f$ on $Sem(SP)$ that contains all pairs $([f(t)],[u])$ with $Sem(SP) \models r_f(t,u)$.

In fact, the interpretation of f in $Sem(SP)/\approx_f$ is the least solution of AX_f w.r.t. $\leq$.

Theorem Let $\sigma : \Sigma' \to \Sigma$ be a signature morphism and $\approx$ be a Σ-congruence on $Ini(SP)$. For all $t, t' \in T_{\Sigma'}$,

$$[t] \approx_\sigma [t'] \quad \Longleftrightarrow_{def} \quad [\sigma(t)] \approx [\sigma(t')]$$

defines a Σ'-congruence on $Ini(SP)|_\sigma$.
$gen(Ini(SP)|_\sigma)/\approx_\sigma$ agrees with $gen(Ini(SP)/\approx|_\sigma)$.
Given a predicate $r \in \Sigma'$, $\approx_\sigma$ is compatible with r if $\approx$ is compatible $\sigma(r)$.
Let Σ_σ consist of Σ and a **representation predicate** $rep_s : \sigma(s)$ for each sort (tuple) $s \in \Sigma'$. Let AX_σ consist of AX and a Horn axiom

$$rep(\sigma(f)(x_1, \ldots, x_n)) \;\Leftarrow\; rep(x_1) \wedge \cdots \wedge rep(x_n)$$

for each function symbol $f \in \Sigma'$. Let $SP_\sigma = (\Sigma_\sigma, AX_\sigma)$.

(1) For all Gentzen clauses C over Σ,

$$Ini(SP) \models C \quad \Longleftrightarrow \quad C \in ITh(SP).$$

(2) For all Gentzen clauses C over Σ',

$$\begin{aligned} Ini(SP)|_\sigma \models C &\iff \sigma(C) \in ITh(SP), \\ gen(Ini(SP)|_\sigma) \models C(x) &\iff Ini(SP_\sigma) \models \sigma(C)(x) \Leftarrow rep(x), \\ gen(Ini(SP)/\approx|_\sigma) \models C(x) &\iff Ini(SP_\sigma)/\approx \models \sigma(C)(x) \Leftarrow rep(x) \end{aligned}$$

where x is a tuple consisting of all variables of C.

(3) Let A be a Σ-structure and $\Sigma' \subseteq \Sigma$ such that $\sim^A$ is A-compatible with Σ'. For all Gentzen clauses C over Σ whose premise is a goal set over Σ',

$$A \models C[\sim / \equiv] \quad \text{implies} \quad A/\approx \models C.$$

(4) Let $\Sigma' \subseteq \Sigma$ and A be a reachable Σ'-structure such that for all ground atoms p over Σ', $A \models p$ iff $Ini(SP) \models p$. $A \cong Ini(SP)_{\Sigma'}$ and for all Gentzen clauses C over Σ',

$$A \models C \quad \Longleftrightarrow \quad Ini(SP) \models C.$$

$$finite\_or\_fair(s) \quad \Rightarrow \quad isnil(s) \ \vee \ \exists\, x, s' : s \xrightarrow{x} s' \ \text{ where}$$

$$finite\_or\_fair(c(\ldots, s, \ldots)) \quad \Leftarrow \quad finite\_or\_fair(s)$$

$$\text{for all } stream\text{-constructors } c \neq filter$$

$$finite\_or\_fair(filter(g, s)) \quad \Leftarrow \quad finite(s) \wedge finite\_or\_fair(s)$$

$$finite\_or\_fair(filter(g, s)) \quad \Leftarrow \quad fair(g, s) \wedge finite\_or\_fair(s)$$

Proof uses inductive expansion with the conjecture along the well-founded relation $\gg$ specified by:

$$c(\ldots, s, \ldots) \gg s \qquad\qquad \text{for all } stream\text{-constructors } c \neq filter$$

$$filter(g, s) \gg filter(g, t) \quad \Leftarrow \quad length(s, m) \wedge length(t, n) \wedge m > n$$

$$filter(g, s) \gg filter(g, t) \quad \Leftarrow \quad first(g, s, m) \wedge first(g, t, n) \wedge m > n$$