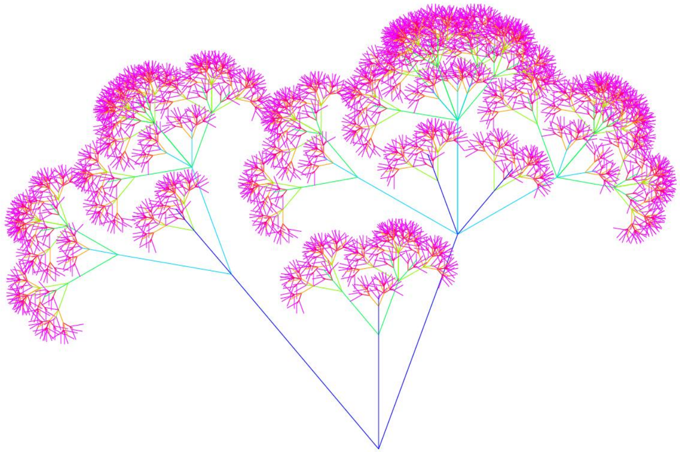
Expander2:

program verification between interaction and automation

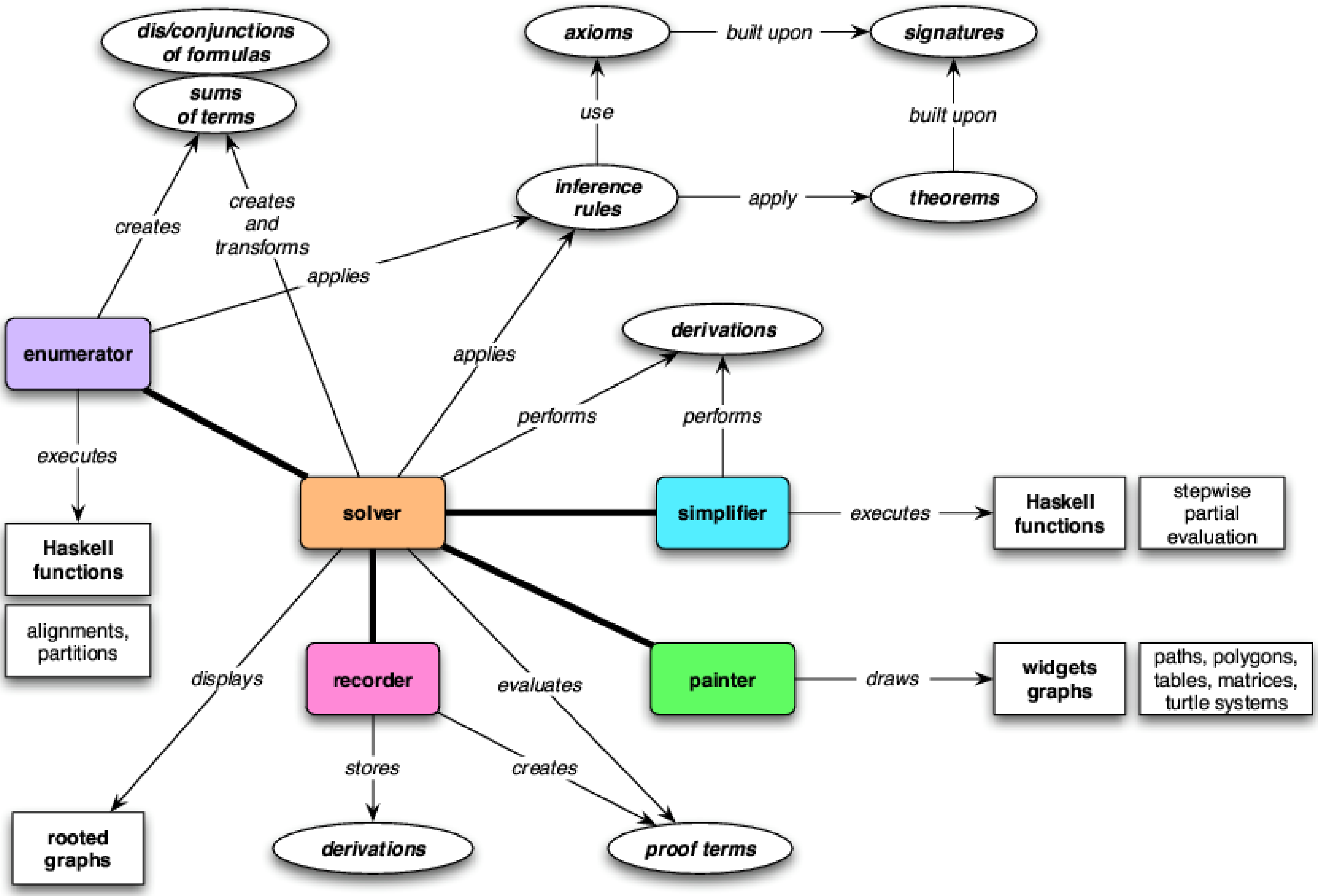
Peter Padawitz
TU Dortmund

February 3, 2021



Contents

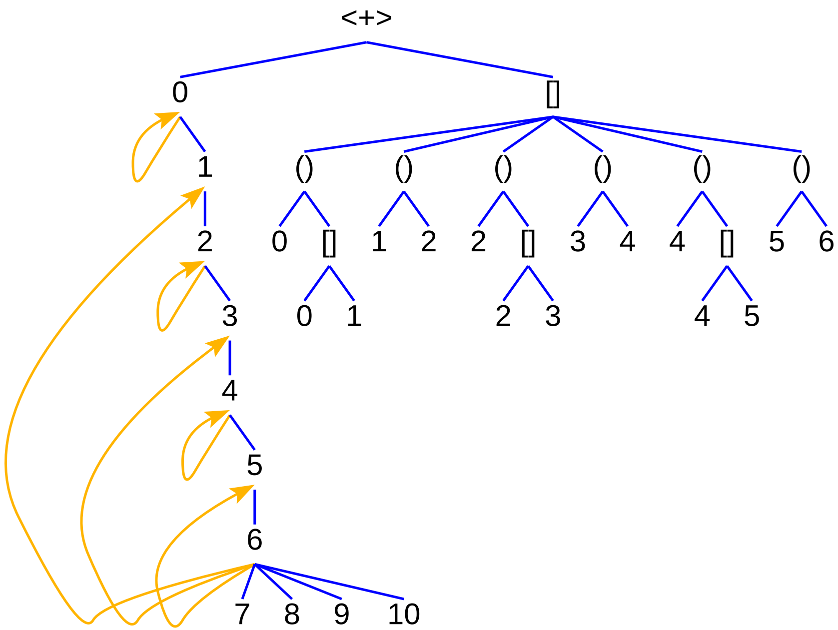
Components of Expander2



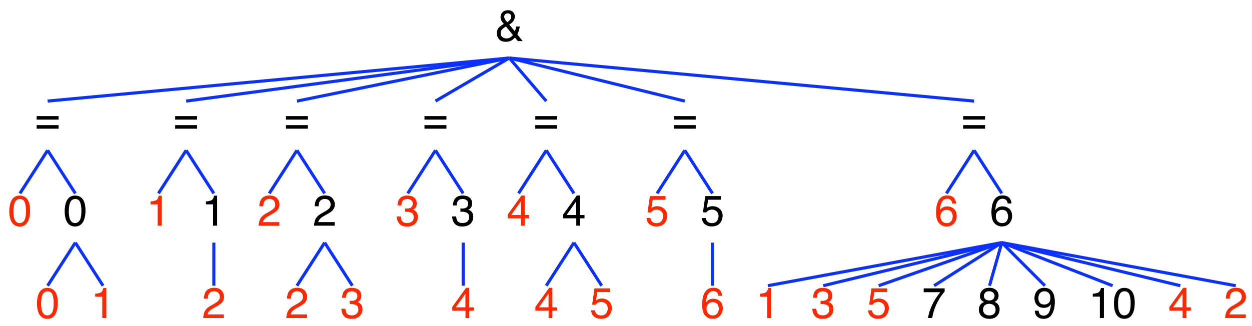
- *3 representations of a formula/term:*
  textual, **tree-like** (tree with additional forward or backward egdes) and **pictorial** (list of 2-dimensional widgets).
  Terms may involve constants denoting **state variables** whose—usually hidden—values can be generated by equational axioms or an enumerator, modified by the simplifier and drawn by the painter. State variables are used for testing iterative algorithms.
  All representations can be edited, moved and scaled.
  Sets of pictorial representations can also be rotated and completed to graphs by adding arcs of different shapes.

Example NDA (TRANS0)

```
defuncts:   states
fovars:     n
axioms:     states = [0..10]                                   &
            (n < 6 & n `mod` 2 = 0 ==> n -> [n,n+1])           &
            (n < 6 & n `mod` 2 =/= 0 ==> n -> n+1)             &
            6 -> [1,3,5,7..10]
```



	0	1	2	3	4	5	6
0	●	●					
1			●				
2			●	●			
3					●		
4					●	●	
5							●

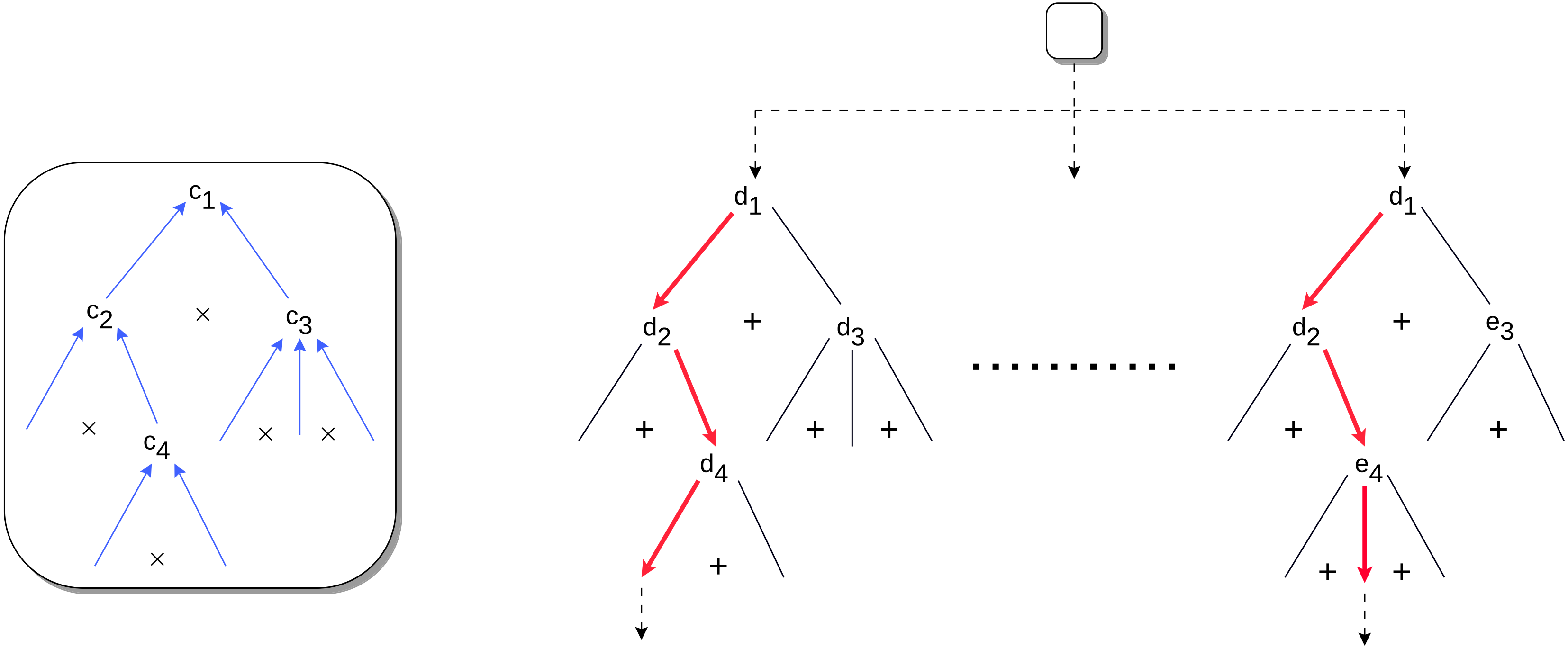


Types and functions

- **Sums** $\coprod_{i\in I} t_i$ formalize case analysis and type extension.
  Products $\prod_{i\in I} t_i$ formalize/implement tupling and type restriction.
- A recursively defined type T is created from

constructors $polyType(T) \xrightarrow{c} T$ represent	or	**destructors** $T \xrightarrow{d} polyType(T)$ represent
data generated by, context-free languages, "finite", "inductive" data		transition systems, "non-deterministic" functions, streams and other "infinite" data
initial T-algebra		final T-coalgebra

- Functions $T \xrightarrow{f} polyType(T)$ are defined as initial catamorphisms (**recursion**).
  Functions $polyType(T) \xrightarrow{g} T$ are defined by as final anamorphisms (**corecursion**).



An element of an initial algebra and a final coalgebra, respectively

Relations, quotients and substructures

- **Horn clauses** $r(t) \Leftarrow \varphi$ define a **predicate (least relation)**
  (safety predicate, transition system)
  as the least solution of $r(t) \Leftarrow \varphi$ in r.
  Co-Horn clauses $r(t) \Rightarrow \varphi$ define a **copredicate (greatest relation)**
  (liveness predicate, non-inductive property)
  as the greatest solution of $r(t) \Rightarrow \varphi$ in r.
- The **complement** of a predicate/copredicate is a copredicate/predicate.
- Properties of least relations are proved by **induction**.
  Properties of greatest relations are proved by **coinduction**.
- Least/greatest **congruences** $\equiv : t \times t$ and **quotients** $A/{\equiv^A}$
  formalize visible/hidden abstraction.
  Least/greatest **invariants** $all : t$ and **substructures** $all^A \subseteq A$
  formalize visible/hidden restriction.

3 specifications

Example Partitioning and flattening finite lists (LIST and LISTEVAL)

```
constructs: [] :
defuncts:   flatten
preds:      part
fovars:     x y s s' p
axioms:     part([x],[[x]])                          &
            (part(x:y:s,[x]:p) <=== part(y:s,p))     &
            (part(x:y:s,(x:s'):p) <=== part(y:s,s':p))   &
            flatten[] = []                           &
            flatten(s:p) = s++flatten(p)
```

Example Streams (infinite lists) (STREAM)

```
specs:       NAT BOOL
constructs:  [] :
defuncts:    head tail eq blink
preds:       exists
copreds:     fair
fovars:      x y s
hovars:      f
axioms:      head(x:s) = x                                               &
             tail(x:s) = s                                               &
             head(blink) = 0                                             &
             tail(blink) = 1:blink                                       &
             eq(x)(x) = true                                             &
             (x =/= y ==> eq(x)(y) = false)                              &
             (f(head(s)) = true  ==> exists(f)(s))                       &
             (f(head(s)) = false
               ==> (exists(f)(s) <=== exists(f)(tail(s))))               &
             (fair(f)(s) ===> exists(f)(s) & fair(f)(tail(s)))
```

Example Model checking (CTLlab)

```
constructs: a b
preds:      P true OD Y ->
copreds:    false OB X
fovars:     x st st'
hovars:     P
axioms:     true(st)                                                 &
            (false(st) ===> False)                                   &
            (OD(x)(P)(st) <=== (st,x) -> st' & P(st'))               &
            (OB(x)(P)(st) ===> ((st,x) -> st' ==> P(st')))           &
            (X(st) ===> Y(st))                                       &
            (X(st) ===> OB(b)(X)(st))                                &
            (Y(st) <=== OD(a)(true)(st))                             &
            (Y(st) <=== OD(b)(Y)(st))                                &
            (2,b) -> 1 & (2,b) -> 3 & (3,b) -> 3 & (3,a) -> 4        &
            (4,b) -> 3
```

Deduction at a glance

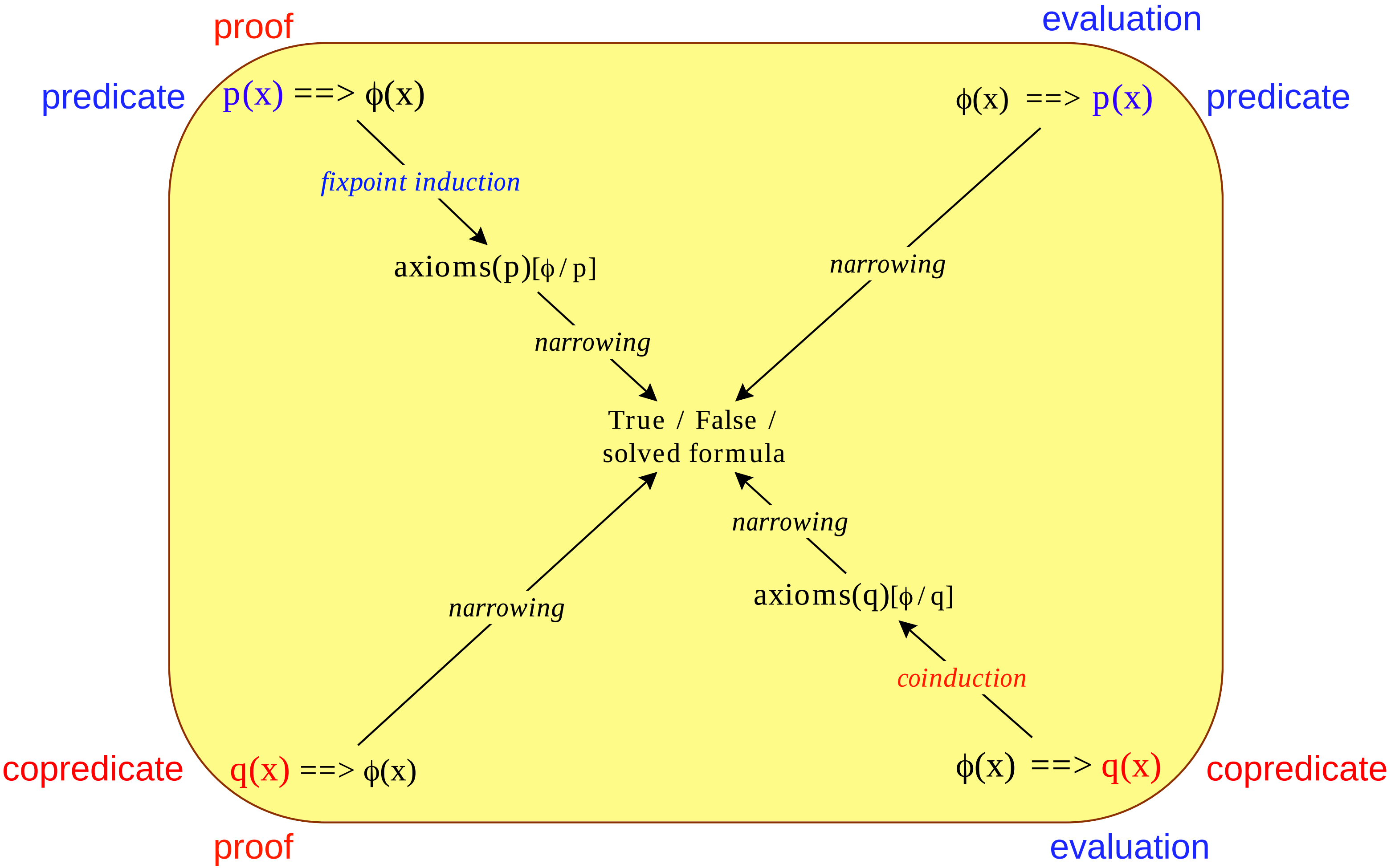
- **Top-down derivations** transform logical formulas into *True* or other solved formulas.

$$
\begin{array}{lll}
\textit{prove } \varphi\text{:} & \varphi & \vdash \quad \mathit{True} \\
\textit{solve } \varphi\text{:} & \varphi & \vdash \quad r(c) \\
\textit{refute } \varphi\text{:} & \neg\varphi & \vdash \quad \mathit{True} \\
\textit{verify } p\text{:} & p(x) \Rightarrow \varphi & \vdash \quad \mathit{True} \\
\textit{evaluate } p\text{:} & p(x) \Leftarrow \varphi & \vdash \quad \mathit{True} \\
\textit{evaluate } t\text{:} & t \equiv x & \vdash \quad c \equiv x
\end{array}
$$

- **Rewrite sequences** generate/modify/normalize terms.

- Rules at three levels of automation/interaction:
  - *bottom:* **Simplifications** are equivalence transformations that partially evaluate terms and formulas.
  - *medium:* **Narrowing** and **rewriting** apply **axioms** to **goals**,
    exhaustively or selectively, interactively or automatically, stepwise or iteratively.
  - *top:* **Induction** and **coinduction** and other proper **expansions** are applied interactively and stepwise.
    Induction and coinduction apply **goals** (hypotheses) to **axioms** and that way prove the former by solving the latter.

Duality of narrowing and co/induction



Narrowing applies axioms to conjectures.
The proof proceeds by transforming the modified conjectures.
Coinduction and fixpoint induction apply conjectures to axioms.
The proof proceeds by transforming the modified axioms.

Narrowing on a predicate p is a rule for evaluating p.
Fixpoint induction on p is a rule for verifying p.

Narrowing on a copredicate q is a rule for verifying q.
Coinduction on q is a rule for evaluating q.

Can co/induction be lifted to the medium level of automation?

Deduction in more detail

Analytical, top-down proofs of φ_1 like

$$\varphi_1 \quad \vdash \quad \varphi_2 \quad \vdash \quad \ldots \quad \vdash \quad \varphi_n$$

are sound (w.r.t. the initial/final specification model):

$$\varphi_1 \quad \Leftarrow \quad \varphi_2 \quad \Leftarrow \quad \ldots \quad \Leftarrow \quad \varphi_n$$

Deterministic or non-deterministic rewritings of t_1:

$$t_1 \quad \rightarrow \quad t_{21}{<}{+}{>}\ldots{<}{+}{>}t_{2k_2} \quad \rightarrow \quad \ldots \quad \rightarrow \quad t_{n1}{<}{+}{>}\ldots{<}{+}{>}t_{nk_n}$$

The term () denotes *undefined*.

prove φ_1:	$\varphi_n = \mathit{True}$
refute φ_1:	$\varphi_n = \mathit{False}$
solve φ_1:	φ_n has the form
	$\exists\ Z_1 : x_1 = c_1 \wedge \cdots \wedge \exists\ Z_k : x_k = c_k \wedge$
	$\forall\ Z_{k+1} : x_{k+1} \neq c_{k+1} \wedge \cdots \wedge \forall\ Z_n : x_n \neq c_n$,
	$x_1, \ldots, x_n$ are different free variables,
	$c_1, \ldots, c_n$ consist of constructors,
	$\{(i, j) \mid c_i \text{ contains } x_j\}^+$ is acyclic.
evaluate t:	$\varphi_1 = (t = x)$, $\varphi_n = (x = c)$
rewrite t *logically*:	$\varphi_1 = (t \rightarrow x)$, $\varphi_n = (x = c1) \wedge \cdots \wedge (x = ck)$

3 levels of automation $\leadsto$ 3 kinds of rules

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

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

- An *expansion* $\dfrac{\varphi}{\psi} \Uparrow$ 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 $C[\varphi]$:

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

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

- The **simplifier** simplifies logical formulas and partially evaluates terms w.r.t. built-in types.
- **Narrowing** applies axioms to formulas. **Rewriting** applies axioms to terms.
  If all applicable axioms are applied at a given position, then narrowing is an equivalence transformation and thus a simplification!
- **Induction**, **coinduction** and other **expansions** are applied locally and stepwise.

The bottom level: Simplifications

- **Elimination of zeros and ones**

$$\frac{\varphi \wedge \mathit{True}}{\varphi} \qquad \frac{\varphi \vee \mathit{False}}{\varphi} \qquad \frac{\varphi \wedge \mathit{False}}{\mathit{False}} \qquad \frac{\varphi \vee \mathit{True}}{\mathit{True}} \qquad \frac{() \to t}{\mathit{False}} \qquad \frac{t \mathrel{<\!+\!>} ()}{t}$$

- **Flattening**

$$\frac{\varphi \wedge (\psi_1 \wedge \cdots \wedge \psi_n)}{\varphi \wedge \psi_1 \wedge \cdots \wedge \psi_n} \qquad \frac{\varphi \vee (\psi_1 \vee \cdots \vee \psi_n)}{\varphi \vee \psi_1 \vee \cdots \vee \psi_n}$$

- **Disjunctive normal form** Let f be a function and p be a co/predicate.

$$\frac{f(\ldots, t_1 \mathrel{<\!+\!>} \ldots \mathrel{<\!+\!>} t_n, \ldots)}{f(\ldots, t_1, \ldots) \mathrel{<\!+\!>} \ldots \mathrel{<\!+\!>} f(\ldots, t_n, \ldots)}$$

$$\frac{p(\ldots, t_1 \mathrel{<\!+\!>} \ldots \mathrel{<\!+\!>} t_n, \ldots)}{p(\ldots, t_1, \ldots) \vee \cdots \vee p(\ldots, t_n, \ldots)}$$

$$\frac{\varphi \wedge \forall \vec{x}(\psi_1 \vee \cdots \vee \psi_n)}{\forall \vec{x}((\varphi \wedge \psi_1) \vee \cdots \vee (\varphi \wedge \psi_n))} \quad \text{if no } x \in \vec{x} \text{ occurs freely } \varphi$$

- **Term decomposition** Let c and d be different constructors.

$$\frac{c(t_1,\ldots,t_n) = c(u_1,\ldots,u_n)}{t_1 = u_1 \wedge \cdots \wedge t_n = u_n} \qquad \frac{c(t_1,\ldots,t_n) = d(u_1,\ldots,u_n)}{\mathit{False}}$$

$$\frac{c(t_1,\ldots,t_n) \neq c(u_1,\ldots,u_n)}{t_1 \neq u_1 \vee \cdots \vee t_n \neq u_n} \qquad \frac{c(t_1,\ldots,t_n) \neq d(u_1,\ldots,u_n)}{\mathit{True}}$$

- **Quantifier distribution**

$$\frac{\forall\vec{x}(\varphi_1 \wedge \cdots \wedge \varphi_n)}{\forall\vec{x}\varphi_1 \wedge \cdots \wedge \forall\vec{x}\varphi_n} \qquad \frac{\exists\vec{x}(\varphi_1 \vee \cdots \vee \varphi_n)}{\exists\vec{x}\varphi_1 \vee \cdots \vee \exists\vec{x}\varphi_n} \qquad \frac{\exists\vec{x}(\varphi \Rightarrow \psi)}{\forall\vec{x}\varphi \Rightarrow \exists\vec{x}\psi}$$

$$\frac{\exists\vec{x}(\varphi_1 \wedge \cdots \wedge \varphi_n)}{\exists\vec{x_1}\varphi_1 \wedge \cdots \wedge \exists\vec{x_n}\varphi_n} \qquad \frac{\forall\vec{x}(\varphi_1 \vee \cdots \vee \varphi_n)}{\forall\vec{x_1}\varphi_1 \vee \cdots \vee \forall\vec{x_n}\varphi_n}$$

if $\vec{x} = \vec{x_1} \cup \cdots \cup \vec{x_n}$ and for all $1 \leq i \leq n$, no variable of $\vec{x_i}$ occurs freely in some φ_j, $1 \leq j \leq n$, $j \neq i$.

- **Removal of negation.** Negation symbols are moved to literal positions where they are replaced by complement predicates: $\neg P(t)$ is reduced to $not\_P(t)$, $\neg not\_P(t)$ is reduced to $P(t)$. Co-Horn/Horn axioms for not_P can be generated automatically from Horn/Co-Horn axioms for P.
- **Removal of quantifiers.** Unused **bounded variables** are removed. **Successive quantifiers** are merged.

Subsumption

$$\frac{\varphi \Rightarrow \psi}{True} \qquad \frac{\varphi \Leftrightarrow \psi}{\psi \Rightarrow \varphi} \qquad \frac{\psi \Leftrightarrow \varphi}{\psi \Rightarrow \varphi} \qquad \frac{\varphi \wedge (\psi \Rightarrow \theta)}{\varphi \wedge \theta} \qquad \text{if } \varphi \text{ subsumes } \psi$$

$$\frac{\varphi_1 \vee \cdots \vee \varphi_n}{\varphi_1 \vee \cdots \vee \varphi_{n-1}} \qquad \text{if } \varphi_1 \wedge \cdots \wedge \varphi_{n-1} \text{ subsumes } \varphi_n$$

$$\frac{\varphi_1 \vee \cdots \vee \varphi_n}{\varphi_1 \vee \cdots \vee \varphi_{n-1}} \qquad \text{if } \varphi_n \text{ subsumes } \varphi_1 \vee \cdots \vee \varphi_{n-1}$$

Subsumption is the least binary relation on terms and formulas that satisfies the following implications: Let $\sim$ be the syntactic equality of formulas modulo the re-arrangement of arguments of permutative operators and θ denote a renaming of variables.

$$
\begin{aligned}
\varphi \sim \psi &\implies \varphi \text{ subsumes } \psi \\
\varphi \text{ subsumes } \psi &\implies \neg\psi \text{ subsumes } \neg\varphi \\
\varphi' \text{ subsumes } \varphi \text{ and } \psi \text{ subsumes } \psi' &\implies \varphi \Rightarrow \psi \text{ subsumes } \varphi' \Rightarrow \psi' \\
\exists\, 1 \leq i \leq n : \varphi \text{ subsumes } \psi_i &\implies \varphi \text{ subsumes } \psi_1 \vee \cdots \vee \psi_n \\
\forall\, 1 \leq i \leq n : \varphi_i \text{ subsumes } \psi &\implies \varphi_1 \vee \cdots \vee \varphi_n \text{ subsumes } \psi \\
\forall\, 1 \leq i \leq n : \varphi \text{ subsumes } \psi_i &\implies \varphi \text{ subsumes } \psi_1 \wedge \cdots \wedge \psi_n \\
\exists\, 1 \leq i \leq n : \varphi_i \text{ subsumes } \psi &\implies \varphi_1 \wedge \cdots \wedge \varphi_n \text{ subsumes } \psi \\
\varphi(\vec{x}) \text{ subsumes } \psi(\vec{x}) &\implies \exists \vec{x} \varphi(\vec{x}) \text{ subsumes } \exists \vec{y} \psi(\vec{y}) \\
\varphi(\vec{x}) \text{ subsumes } \psi(\vec{x}) &\implies \forall \vec{x} \varphi(\vec{x}) \text{ subsumes } \forall \vec{y} \psi(\vec{y}) \\
\exists\, \theta, \vec{t} : \varphi\theta \sim \psi(\vec{t}) &\implies \varphi \text{ subsumes } \exists \vec{x} \psi(\vec{x}) \\
\exists\, \theta, \vec{t} : \varphi(\vec{t}) \sim \psi\theta &\implies \forall \vec{x} \varphi(\vec{x}) \text{ subsumes } \psi \\
\exists\, \theta, \vec{t}, \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\} : (\varphi_{i_1} \wedge \cdots \wedge \varphi_{i_k})\theta \sim \psi(\vec{t}) & \\
&\implies \varphi_1 \wedge \cdots \wedge \varphi_n \text{ subsumes } \exists \vec{x} \psi(\vec{x}) \\
\exists\, \theta, \vec{t}, \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\} : \varphi(\vec{t}) \sim (\psi_{i_1} \vee \cdots \vee \psi_{i_k})\theta & \\
&\implies \forall \vec{x} \varphi(\vec{x}) \text{ subsumes } \psi_1 \vee \cdots \vee \psi_n
\end{aligned}
$$

- **Elimination of equations and inequations.** Let $x \in \vec{x} \setminus var(t)$.

$$\frac{\exists \vec{x}(x = t \wedge \varphi)}{\exists \vec{x} \varphi[t/x]} \qquad \frac{\forall \vec{x}(x \neq t \vee \varphi)}{\forall \vec{x} \varphi[t/x]}$$

$$\frac{\forall \vec{x}(x = t \wedge \varphi \Rightarrow \psi)}{\forall \vec{x}(\varphi \Rightarrow \psi)[t/x]} \qquad \frac{\forall \vec{x}(\varphi \Rightarrow x \neq t \vee \psi)}{\forall \vec{x}(\varphi \Rightarrow \psi)[t/x]}$$

- **Substitution by normal forms.** Let $x \in \vec{x} \setminus var(t)$ and t be a normal form.

$$\frac{\exists \vec{x}(x = t \wedge \varphi)}{\exists \vec{x}(x = t \wedge \varphi[t/x])} \qquad \frac{\forall \vec{x}(x \neq t \vee \varphi)}{\forall \vec{x}(x \neq t \vee \varphi[t/x])}$$

$$\frac{\forall \vec{x}(x = t \wedge \varphi \Rightarrow \psi)}{\forall \vec{x}(x = t \wedge \varphi[t/x] \Rightarrow \psi[t/x])} \qquad \frac{\forall \vec{x}(\varphi \Rightarrow x \neq t \vee \psi)}{\forall \vec{x}(\varphi[t/x] \Rightarrow x \neq t \vee \psi[t/x])}$$

- **Universal quantification of implications**

$$\frac{\exists\vec{x}\varphi \Rightarrow \psi}{\forall\vec{x}(\varphi \Rightarrow \psi)} \qquad \frac{\psi \Rightarrow \forall\vec{x}\varphi}{\forall\vec{x}(\psi \Rightarrow \varphi)}$$

  if no variable of $\vec{x}$ occurs freely in ψ.

- **Implication splitting**

$$\frac{\forall\vec{x}(\varphi_1 \vee \cdots \vee \varphi_n \Rightarrow \psi)}{\forall\vec{x}(\varphi_1 \Rightarrow \psi) \wedge \cdots \wedge \forall\vec{x}(\varphi_n \Rightarrow \psi)} \qquad \frac{\forall\vec{x}(\varphi \Rightarrow \psi_1 \wedge \cdots \wedge \psi_n)}{\forall\vec{x}(\varphi \Rightarrow \psi_1) \wedge \cdots \wedge \forall\vec{x}(\varphi \Rightarrow \psi_n)}$$

- **Uncurrying**

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

The medium level: narrowing and rewriting

Axioms and theorems are **Horn clauses** ((1)-(7)), **co-Horn clauses** ((8)-(12)) or **tautologies** ((13) and (14)).

Let f be a defined function, p be a predicate, q be a copredicate and $at_1, \dots, at_n$ be atoms.

$$
\begin{array}{llll}
(1) & \{guard \Rightarrow\} & (\underline{f(\vec{t})} = u & \{\Longleftarrow prem\}) \\
(2) & \{guard \Rightarrow\} & (\underline{t_1 \wedge \ldots \wedge t_n} \rightarrow u & \{\Longleftarrow prem\}) \\
(3) & \{guard \Rightarrow\} & (\underline{p(\vec{t})} & \{\Longleftarrow prem\}) \\
(4) & & \underline{t} = u & \{\Longleftarrow prem\} \\
(5) & & \underline{q(\vec{t})} & \{\Longleftarrow prem\} \\
(6) & & \underline{at_1} \wedge \cdots \wedge \underline{at_n} & \{\Longleftarrow prem\} \\
(7) & & \underline{at_1} \vee \cdots \vee \underline{at_n} & \{\Longleftarrow prem\} \\
(8) & \{guard \Rightarrow\} & (\underline{q(\vec{t})} & \Longrightarrow conc) \\
(9) & & \underline{t} = u & \Longrightarrow conc \\
(10) & & \underline{p(\vec{t})} & \Longrightarrow conc \\
(11) & & \underline{at_1} \wedge \cdots \wedge \underline{at_n} & \Longrightarrow conc \\
(12) & & \underline{at_1} \vee \cdots \vee \underline{at_n} & \Longrightarrow conc \\
(13) & & conc & \\
(14) & & \neg prem &
\end{array}
$$

narrowing upon a predicate p

$$\frac{p(t)}{\bigvee_{i=1}^{k} \exists Z_i : (\varphi_i\sigma_i \wedge \vec{x} = \vec{x}\sigma_i)}$$

where $\gamma_1 \Rightarrow (p(t_1) \Longleftarrow \varphi_1), \ldots, \gamma_n \Rightarrow (p(t_n) \Longleftarrow \varphi_n)$ are the axioms for p,

$(*)$ $\vec{x}$ is a list of the variables of t,
for all $1 \leq i \leq k$, $t\sigma_i = t_i\sigma_i$, $\gamma_i\sigma_i \vdash \mathit{True}$ and $Z_i = \mathit{var}(t_i, \varphi_i)$,
for all $k < i \leq n$, t is not unifiable with t_i.

narrowing upon a copredicate p

$$\frac{p(t)}{\bigwedge_{i=1}^{k} \forall Z_i : (\varphi_i\sigma_i \vee \vec{x} \neq \vec{x}\sigma_i)}$$

where $\gamma_1 \Rightarrow (p(t_1) \Longrightarrow \varphi_1), \ldots, \gamma_n \Rightarrow (p(t_n) \Longrightarrow \varphi_n)$ are the axioms for p and $(*)$ holds true.

narrowing upon a defined function f

$$\frac{r(\ldots,f(t),\ldots)}{\begin{array}{c}\bigvee_{i=1}^{k}\exists Z_i:(r(\ldots,u_i,\ldots)\sigma_i\wedge\varphi_i\sigma_i\wedge\vec{x}=\vec{x}\sigma_i)\vee\\ \bigvee_{i=k+1}^{l}(r(\ldots,f(t),\ldots)\sigma_i\wedge\vec{x}=\vec{x}\sigma_i)\end{array}}$$

where r is a predicate or copredicate,

$\gamma_1\Rightarrow(f(t_1)=u_1\Longleftarrow\varphi_1),\ldots,\gamma_n\Rightarrow(f(t_n)=u_n\Longleftarrow\varphi_n)$ are the axioms for f,

(∗∗) $\vec{x}$ is a list of the variables of t,
for all $1\leq i\leq k$, $t\sigma_i=t_i\sigma_i$, $\gamma_i\sigma_i\vdash\mathit{True}$ and $Z_i=var(t_i,\varphi_i)$,
for all $k<i\leq l$, σ_i is a partial unifier of t and t_i,
for all $l<i\leq n$, t is not partially unifiable with t_i.

narrowing upon the predicate $\rightarrow$

$$\frac{t\ {}^\wedge v\rightarrow t'}{\begin{array}{c}\bigvee_{i=1}^{k}\exists Z_i:((u_i\ {}^\wedge v)\sigma_i=t'\sigma_i\wedge\varphi_i\sigma_i\wedge\vec{x}=\vec{x}\sigma_i)\vee\\ \bigvee_{i=k+1}^{l}((t\ {}^\wedge v)\sigma_i\rightarrow t'\sigma_i\wedge\vec{x}=\vec{x}\sigma_i)\end{array}}$$

where $\gamma_1\Rightarrow(t_1\rightarrow u_1\Longleftarrow\varphi_1),\ldots,\gamma_n\Rightarrow(t_n\rightarrow u_n\Longleftarrow\varphi_n)$ are the axioms for $\rightarrow$, (∗∗) holds true and σ_i is a unifier *modulo associativity and commutativity of* $^\wedge$

elimination of non-narrowable atoms and terms

$$\frac{p(t)}{False} \qquad \frac{q(t)}{True} \qquad \frac{r(\dots,f(t),\dots)}{r(\dots,(),\dots)} \qquad \frac{t\rightarrow t'}{()\rightarrow t'}$$

where $p \neq \rightarrow$ is a predicate, q is a copredicate, r is a predicate or copredicate, f is a defined function, t is a normal form and for all axioms $\gamma \Rightarrow (p(u) \Longleftarrow \varphi)$, $\gamma \Rightarrow (q(u) \Longrightarrow \varphi)$, $\gamma \Rightarrow (f(u) = v \Longleftarrow \varphi)$ and $\gamma \Rightarrow (u \rightarrow v \Longleftarrow \varphi)$, t and u are not unifiable.

rewriting upon a defined function f

$$\frac{c(f(t))}{c(u_1\sigma_1){<}{+}{>}\ldots{<}{+}{>}c(u_k\sigma_k)}$$

where $\gamma_1 \Rightarrow f(t_1) = u_1, \ldots, \gamma_1 \Rightarrow f(t_n) = u_n$ are the axioms for f and

$(*)$ for all $1 \leq i \leq k$, $t = t_i\sigma_i$ and $\gamma_i\sigma_i \vdash \mathit{True}$,
for all $k < i \leq n$, t does not match t_i.

rewriting upon the predicate $\rightarrow$

$$\frac{c(t)}{c(u_1\sigma_1){<}{+}{>}\ldots{<}{+}{>}c(u_k\sigma_k)}$$

where $\gamma_1 \Rightarrow t_1 \rightarrow u_1, \ldots, \gamma_1 \Rightarrow t_n \rightarrow u_n$ are the axioms for $\rightarrow$ and $(*)$ holds true.

elimination of non-rewritable terms

$$\frac{f(t)}{()}$$

where f is a defined function, t is a normal form
and for all axioms $\gamma \Rightarrow f(u) = v$ and $\gamma \Rightarrow u \rightarrow v$, t and u are not unifiable.

The top level: induction and coinduction

- **Noetherian induction.** Select a list of free or universal induction variables $x_1, \ldots, x_n$ in the displayed tree.
  If $\varphi = (prem \Rightarrow conc)$, then the *induction hypotheses*

  $$\begin{aligned} conc' &\Longleftarrow (x_1, \ldots, x_n) \gg (x'_1, \ldots, x'_n) \ \wedge \ prem' \\ prem' &\Longrightarrow ((x_1, \ldots, x_n) \gg (x'_1, \ldots, x'_n) \ \Rightarrow \ conc') \end{aligned}$$

  are added to the current theorems.
  If φ is not an implication, then

  $$conc' \Longleftarrow (x_1, \ldots, x_n) \gg (x'_1, \ldots, x'_n)$$

  is added. Primed formulas are obtained from unprimed ones by priming the occurrences of $x_1, \ldots, x_n$.
  $\gg$ denotes the induction ordering. Each left-to right application of an added theorem corresponds to an induction step and introduces an occurrence of $\gg$.
  After axioms for $\gg$ have been added to the current axioms, narrowing steps upon $\gg$ should remove the occurrences of $\gg$ because the transformation is correct only if φ can be derived to *True*.

The following rules are correct if the selected subformulas have positive polarity.

For each predicate, copredicate or function p, let AX_p be the set of axioms for p.

- **coinduction on a copredicate** p

$$\frac{\psi(x) \;\Rightarrow\; p(x)}{\bigwedge_{p(t)\Rightarrow\varphi\in AX_p}(\psi(t) \Rightarrow \varphi[\psi/p])} \Uparrow$$

Realization:

➢ Select subformulas

$$\begin{array}{ll} & \{prem_1 \Rightarrow\}\ p(\vec{t_1}) \\ \wedge & \ldots \\ \wedge & \{prem_k \Rightarrow\}\ p(\vec{t_k}) \end{array} \qquad \text{(A)}$$

such that p does not depend on any predicate or function occurring in $prem_i$.

(A) is turned into

$$
\begin{aligned}
p(\vec{x}) \quad \Longleftarrow \quad & \{prem_1 \wedge\} \; \vec{x} = \vec{t_1} \\
\vee \; & \ldots \\
\vee \; & \{prem_k \wedge\} \; \vec{x} = \vec{t_k}
\end{aligned}
\qquad \text{(A')}
$$

where $\vec{x}$ is a list of variables.

➢ A new predicate p' is added to the current signature and

$$
\begin{aligned}
p'(\vec{x}) \quad \Longleftarrow \quad & \{prem_1 \wedge\} \; \vec{x} = \vec{t_1} \\
\vee \; & \ldots \\
\vee \; & \{prem_k \wedge\} \; \vec{x} = \vec{t_k}
\end{aligned}
\qquad (AX_0)
$$

becomes the axiom for p'.

➢ AX_0 is applied to $AX_p[p'/p]$.

➢ The conjunction of the resulting clauses replaces the original conjecture A.

- **n-level coinduction on a copredicate p**

$$\frac{\psi(x) \Rightarrow p(x)}{\bigwedge_{p(t)\Rightarrow\varphi\in AX_p}(\psi(t) \Rightarrow \varphi'[\psi/p])} \Uparrow$$

where $\varphi \vdash^n_{AX_p} \varphi'$.

Realization:

- ➢ Select subformulas of the form A and turn them into A'.
- ➢ A new predicate p' is added to the current signature and AX_0 becomes the axiom for p'.
- ➢ For all $p(t) \Rightarrow \varphi \in AX_p$, let φ' be the result of submitting φ to a sequence of n inference steps each of which consists of the parallel application of AX_p to all current redices.
- ➢ AX_0 is applied to $p'(t) \Rightarrow \varphi'[p'/p]$.
- ➢ The conjunction of the resulting clauses replaces the original conjecture A.

- **fixpoint induction on a predicate p**

$$\frac{p(x) \;\Rightarrow\; \psi(x)}{\bigwedge_{p(t)\Leftarrow\varphi\in AX_p}(\varphi[\psi/p] \Rightarrow \psi(t))} \Uparrow$$

Realization:

➢ Select subformulas

$$\begin{array}{ll} & p(\vec{t_1}) \;\Rightarrow\; conc_1 \\ \wedge & \ldots \\ \wedge & p(\vec{t_k}) \;\Rightarrow\; conc_k \end{array} \qquad \text{(B)}$$

such that p does not depend on any predicate or function occurring in $conc_i$. (B) is turned into

$$\begin{array}{lll} p(\vec{x}) & \Longrightarrow & \quad (\vec{x} = \vec{t_1} \;\Rightarrow\; conc_1) \\ & & \wedge \;\; \ldots \\ & & \wedge \;\; (\vec{x} = \vec{t_k} \;\Rightarrow\; conc_k) \end{array} \qquad \text{(B')}$$

where $\vec{x}$ is a list of variables.

- A new predicate p' is added to the current signature and

$$\begin{aligned} p'(\vec{x}) \quad \Longrightarrow \quad & \quad (\vec{x} = \vec{t_1} \;\Rightarrow\; conc_1) \\ & \wedge \;\; \ldots \\ & \wedge \;\; (\vec{x} = \vec{t_k} \;\Rightarrow\; conc_k) \end{aligned} \qquad (AX_0)$$

  becomes the axiom for p'.
- AX_0 is applied to $AX_p[p'/p]$.
- The conjunction of the resulting clauses replaces the original conjecture B.

- **n-level fixpoint induction on a predicate p**

$$\frac{p(x) \Rightarrow \psi(x)}{\bigwedge_{p(t)\Leftarrow\varphi\in AX_p}(\varphi'[\psi/p] \Rightarrow \psi(t))} \Uparrow$$

where $\varphi \vdash^n_{AX_p} \varphi'$.

Realization:

➢ Select subformulas of the form B and turn them into B'.

➢ A new predicate p' is added to the current signature and AX_0 becomes the axiom for p'.

➢ For all $p(t) \Leftarrow \varphi \in AX_p$, let φ' be the result of submitting φ to a sequence of n inference steps each of which consists of the parallel application of AX_p to all current redices.

➢ AX_0 is applied to $\varphi'[p'/p] \Rightarrow p'(t)$.

➢ The conjunction of the resulting clauses replaces the original conjecture B.

- **fixpoint induction on a function f**

$$\frac{f(x) \equiv y \;\Rightarrow\; \psi(x,y)}{\bigwedge_{f(t)\equiv u \Leftarrow \varphi \in flat(AX_f)} (\varphi[\psi/(f(\_) \equiv \_)] \Rightarrow \psi(t,u))} \Uparrow$$

Realization:

➢ Select subformulas

$$\begin{aligned} & f(\vec{t_1}) = u_1 \;\Rightarrow\; conc_1 \\ \wedge\; & \ldots \\ \wedge\; & f(\vec{t_k}) = u_k \;\Rightarrow\; conc_k \end{aligned} \tag{C}$$

or

$$\begin{aligned} & f(\vec{t_1}) = u_1 \; \{\wedge\; conc_1\} \\ \wedge\; & \ldots \\ \wedge\; & f(\vec{t_k}) = u_k \; \{\wedge\; conc_k\} \end{aligned} \tag{D}$$

such that f does not depend on any predicate or function occurring in u_i or $conc_i$.

(C) is turned into

$$
\begin{aligned}
f(\vec{x}) = z \quad \Longrightarrow \quad & (\vec{x} = \vec{t_1} \wedge \ z = u_1 \ \Rightarrow \ conc_1) \\
\wedge \ & \ldots \\
\wedge \ & (\vec{x} = \vec{t_k} \wedge \ z = u_k \ \Rightarrow \ conc_k),
\end{aligned} \tag{C'}
$$

(D) is turned into

$$
\begin{aligned}
f(\vec{x}) = z \quad \Longrightarrow \quad & (\vec{x} = \vec{t_1} \ \Rightarrow \ z = u_1\{\wedge \ conc_1\}) \\
\wedge \ & \ldots \\
\wedge \ & (\vec{x} = \vec{t_k} \ \Rightarrow \ z = u_k\{\wedge \ conc_k\})
\end{aligned} \tag{D'}
$$

where $\vec{x}$ is a list of variables and z is a variable.

- A new predicate f' is added to the current signature and

$$f'(\vec{x}, z) \quad \Longrightarrow \quad \begin{array}{ll} & ((\vec{x} = \vec{t_1} \land z = t_1) \;\Rightarrow\; conc_1) \\ \land & \ldots \\ \land & ((\vec{x} = \vec{t_k} \land z = t_k) \;\Rightarrow\; conc_k) \end{array} \qquad (AX_0)$$

  resp.

$$f'(\vec{x}, z) \quad \Longrightarrow \quad \begin{array}{ll} & (\vec{x} = \vec{t_1} \;\Rightarrow\; (z = t_1\{\land\, conc_1\})) \\ \land & \ldots \\ \land & (\vec{x} = \vec{t_k} \;\Rightarrow\; (z = t_k\{\land\, conc_k\})) \end{array} \qquad (AX_0)$$

  becomes the axiom for f'.
- AX_0 is applied to $flat(AX_f)[f'/(f(\_) \equiv \_)]$.
- The conjunction of the resulting clauses replaces the original conjecture C/D.

- **n-level fixpoint induction on a function f**

$$\frac{f(x) \equiv y \Rightarrow \psi(x,y)}{\bigwedge_{f(t)\equiv u \Leftarrow \varphi \in flat(AX_f)} (\varphi'[\psi/(f(\_) \equiv \_)] \Rightarrow \psi(t,u))} \Uparrow$$

where $\varphi \vdash^n_{flat(AX_f)} \varphi'$.

Realization:

➢ Select subformulas of the form C/D and turn them into C'/D'.

➢ A new predicate f' is added to the current signature and AX_0 becomes the axiom for f'.

➢ For all $f(t) \equiv u \Leftarrow \varphi \in flat(AX_f)$, let φ' be the result of submitting φ to a sequence of n inference steps each of which consists of the parallel application of $flat(AX_f)$ to all current redices.

➢ AX_0 is applied to $\varphi'[f'/(f(\_) \equiv \_)] \Rightarrow f'(t,u)$.

➢ The conjunction of the resulting clauses replaces the original conjecture C/D.

- **Hoare induction.** Select a subformula of the form

$$f(t_1, \ldots, t_n) = t \ \Rightarrow \ conc \qquad \text{(A)}$$

or

$$f(t_1, \ldots, t_n) = t \ \{\wedge \ conc\} \qquad \text{(B)}$$

such that f is a **derived function**, i.e. f has a single axiom of the form

$$f(x_1, \ldots, x_n) = g(u_1, \ldots, u_k)$$

or, if the term t_i in A/B has been selected (in addition to A/B itself), f has a single axiom of the form

$$f(x_1, \ldots, x_n) = g(x_i, \ldots, x_n, u_1, \ldots, u_k)$$

with distinct variables $x_1, \ldots, x_n$. A is turned into $INV1 \wedge INV2$ where

$$INV(x_1, \ldots, x_n, u_1, \ldots, u_k) \qquad \text{(INV1)}$$

$$\begin{aligned} & g(x_i, \ldots, x_n, y_1, \ldots, y_k) = z \ \wedge \ INV(x_1, \ldots, x_n, y_1, \ldots, y_k) \\ & \qquad \Rightarrow \quad (x_1 = t_1 \ \wedge \ \ldots \ \wedge \ x_n = t_n \ \wedge \ x = t \ \Rightarrow \ conc) \qquad \text{(INV2)} \end{aligned}$$

while B is turned into $INV1 \wedge INV3$ where

$$\begin{aligned} & g(x_i, \ldots, x_n, y_1, \ldots, y_k) = z \ \wedge \ INV(x_1, \ldots, x_n, y_1, \ldots, y_k) \\ & \qquad \Rightarrow \quad (x_1 = t_1 \ \wedge \ \ldots \ \wedge \ x_n = t_n \Rightarrow \ x = t \ \{\wedge \ conc\} \qquad \text{(INV3)} \end{aligned}$$

If t_i has not been selected in A/B, then $g(x_i, \ldots, x_n, y_1, \ldots, y_k)$ reduces to $g(y_1, \ldots, y_k)$. Usually, the proof proceeds by narrowing INV1, shifting

$$INV(x_1, \ldots, x_n, y_1, \ldots, y_k)$$

from the premise to the conclusion of INV2/INV3 and submitting the resulting formula to fixpoint induction.

- **Subgoal induction** works the same as Hoare induction except that a selected conjecture of the form A is turned into $INV1 \wedge INV2$ where

$$INV(x_i, \dots, x_n, u_1, \dots, u_k, z) \qquad\qquad \Rightarrow \quad (x_1 = t_1 \ \wedge \ \dots \ \wedge \ x_n = t_n \ \wedge \ x = t \ \Rightarrow \ conc \qquad \text{(INV1)}$$

$$g(x_i, \dots, x_n, y_1, \dots, y_k) = z \ \Rightarrow \ INV(x_i, \dots, x_n, y_1, \dots, y_k, z) \qquad \text{(INV2)}$$

while a selected conjecture of the form B is turned into $INV1 \wedge INV3$ where

$$g(x_i, \dots, x_n, y_1, \dots, y_k) = z \ \Rightarrow \ INV(x_i, \dots, x_n, y_1, \dots, y_k, z) \qquad \text{(INV3)}$$

Usually, the proof proceeds by narrowing INV1 and submitting INV2/INV3 to fixpoint induction.

3 proofs

Example PARTproof

```
part(s,p) ==> s = flatten(p)

Applying fixpoint induction w.r.t.

  part([x],[[x]])
& (part(x:y:s,[x]:p) <=== part(y:s,p))
& (part(x:y:s,x:s':p) <=== part(y:s,s':p))

at position [] of the preceding formula leads to

All x:([x] = flatten[[x]]) &
All x y s p:(y:s = flatten(p) ==> x:y:s = flatten([x]:p)) &
All x y s p s':(y:s = flatten(s':p) ==> x:y:s = flatten(x:s':p))

The reducts have been simplified.

Applying the axiom resp. theorem
```

flatten(s:p) = s++flatten(p)

at positions [2,0,1,1],[2,0,0,1],[1,0,1,1],[0,0,1] of the preceding formu

[] = flatten[]

The reducts have been simplified.

Narrowing the preceding formula leads to

True

Example FAIRBLINK

```
fair(eq(0))(blink) & fair(eq(0))(1:blink)

Applying coinduction w.r.t.

  (fair(f)(s) ===> exists(f)(s) & fair(f)(tail(s)))

at position [] of the preceding formula leads to

exists(eq(0))(1:blink) & tail(blink) = 1:blink & exists(eq(0))(blink) |
exists(eq(0))(1:blink) & tail(blink) = blink & exists(eq(0))(blink)

The reducts have been simplified.

Narrowing the preceding formula (3 steps) leads to

True
```

Example CTLlabproof

```
X(3) & X(4)

Applying coinduction w.r.t.

  (X(st) ===> Y(st))
& (X(st) ===> OB(b)(X)(st))

at position [] of the preceding formula leads to

Y(3) & Y(4) & OB(b)(X0)(3) & OB(b)(X0)(4)

The reducts have been simplified.

Narrowing the preceding formula (25 steps) leads to

True
```

Ongoing work

- compilers translating Haskell, Maude or Curry programs into simplification rules
- generating certain axioms or lemmas automatically from the given ones like it is already done with
  - axioms for complements
  - lemmas expressing the fixpoint property of a co/predicate
  - Horn axioms derived from co-Horn axioms
  - (Noetherian) induction hypotheses
  - axiom flattening
  - λ-applications derived from axiom premises
- integrating genuinely coalgebraic concepts into Expander2
  - subtypes
  - invariants
- application to the specification and verification of OO programs
- maybe: O'Haskell interface to Java (replacing Tcl/Tk)

The top level: More expansions

- **Instantiation.** Select an existentially/universally quantified variable x.
  If the scope of x has positive/negative polarity, then all occurrences of x in the scope are replaced by the term in the solver's entry field.
  Alternatively, the replacing term t may be taken from the displayed tree and moved to a position of x in the scope. Again, all occurrences of x in the scope are replaced by t.
- **Generalization.** Select a subformula φ and enter a formula ψ into the solver's entry field. If φ has positive/negative polarity, then φ is combined conjunctively/disjunctively with ψ.

- **Vertical shift of quantifiers.** Select quantified arguments of a propositional operator *op*, i.e. $op \in \{\wedge, \vee, \neg, \Rightarrow\}$. The quantifiers are shifted in front of *op* after all bound variables that also occur freely in some argument or in more than one argument of *op* have been renamed. For instance, a clause of type (6) or (11) cannot be applied to existentially quantified factors and a clause of type (7) or (12) cannot be applied to universally quantified summands. Hence moving the quantifiers out of the conjunction resp. disjunction may be necessary.
- **Horizontal shift of subformulas.** Select an implication

$$prem_1 \wedge \cdots \wedge prem_m \quad \Rightarrow \quad conc_1 \vee \cdots \vee conc_n,$$

premises $prem_{i_1}, \ldots, prem_{i_k}$ and/or conclusions $conc_{j_1}, \ldots, conc_{j_l}$. The implication is turned into

$$\begin{array}{l} prem_{i'_1} \wedge \cdots \wedge prem_{i'_r} \wedge \neg conc_{j_1} \wedge \cdots \wedge \neg conc_{j_l} \\ \quad \Rightarrow \quad \neg prem_{i_1} \vee \cdots \vee \neg prem_{i_k} \vee conc_{j'_1} \vee \cdots \vee conc_{j'_s} \end{array}$$

where $i'_1, \ldots, i'_r = \{1, \ldots, m\} \setminus \{i_1, \ldots, i_k\}$ and $j'_1, \ldots, j'_s = \{1, \ldots, n\} \setminus \{j_1, \ldots, j_l\}$. Such a transformation may be necessary if the original implication shall be proved by fixpoint induction or coinduction.

- **Unification.** Select two factors of a conjunction

$$\varphi = \exists \vec{x}(\varphi_1 \wedge \cdots \wedge \varphi_n)$$

with positive polarity or two summands of a disjunction

$$\psi = \forall \vec{x}(\varphi_1 \vee \cdots \vee \varphi_n)$$

with negative polarity.
If they are unifiable and the unifier instantiates only variables of $\vec{x}$, then one of them is removed and the unifier is applied to the remaining conjunction/disjunction.

- **Copy.** Select a subtree ϕ whose parent node holds a conjunction or disjunction symbol. A copy of ϕ is added to the children of the subtree's parent node.
- **Removal.** Select summands/factors $\phi_1, \ldots, \phi_n$ of the same disjunction/conjunction with positive/negative polarity.
  $\phi_1, \ldots, \phi_n$ are removed from the displayed tree.
- **Reversal.** Select subtrees, which are arguments of the same occurrence of a *permutative* operator. Currently, the permutative operators are:

  $$\&, |, =, = / =, \sim, \sim / \sim, +, *, \ ^\wedge, \{\}.$$

  The list of selected subtrees is reversed.

- **Atom decomposition.**

$$\frac{f(t_1,\dots,t_n) = f(u_1,\dots,u_n)}{t_1 = u_1 \wedge \dots \wedge t_n = u_n} \Uparrow \qquad \frac{f(t_1,\dots,t_n) \neq f(u_1,\dots,u_n)}{t_1 \neq u_1 \vee \dots \vee t_n \neq u_n} \Downarrow$$

- **Replacement by other sides.**

$$\frac{t = u \wedge \varphi(t)}{t = u \wedge \varphi(u)} \Updownarrow \qquad \frac{t \neq u \vee \varphi(t)}{t \neq u \vee \varphi(u)} \Updownarrow$$

$$\frac{t = u \wedge \varphi(t) \Rightarrow \psi(t)}{t = u \wedge \varphi(u) \Rightarrow \psi(u)} \Updownarrow \qquad \frac{\varphi(t) \Rightarrow t \neq u \vee \psi(t)}{\varphi(u) \Rightarrow t \neq u \vee \psi(u)} \Updownarrow$$

- **Transitivity.** Select an atom tRt' with positive polarity or $n-1$ factors

$$t_1Rt_2,\ t_2Rt_3,\ \dots,\ t_{n-1}Rt_n$$

of a conjunction with negative polarity such that R is among $<, \leq, >, \geq, =, \sim$. The selected atoms are decomposed resp. composed in accordance with the assumption that R is transitive.

- **Narrowing on particular axioms.** Select subtrees $\phi_1, \ldots, \phi_n$ with positive/negative polarity and write Horn/co-Horn axioms into the text field or a signature symbol f into the solver's entry field.
  Narrowing/rewriting steps upon the axioms in the text field or the axioms for f, respectively, are applied to $\phi_1, \ldots, \phi_n$.
- **Axiom/theorem application.** Select subtrees $\phi_1, \ldots, \phi_n$ with positive/negative polarity and write the number of a Horn/co-Horn axiom or theorem into the solver's entry field.
  The selected axiom or theorem ψ is applied from left to right or from right to left to $\phi_1, \ldots, \phi_n$. Left/right refers to t resp. u if ψ has the form $tRu \Longleftarrow prem$ where R is symmetric and to the formula left/right of $\Longleftarrow$ resp. $\Longrightarrow$ in all other cases. The transformation is correct if the conclusion/premise of ψ has positive/negative polarity.

  A clause of type (6), (7), (11) or (12) is applied to atoms $at'_1, \ldots, at'_n$ each of which is part of a conjunction or disjunction: Let $\vec{z}$ consist of the free variables of $prem$ resp. $conc$ that do not occur in $at_1, \ldots, at_n$.

$$\textit{application of (6)} \quad \frac{\varphi_1(at'_1) \wedge \cdots \wedge \varphi_n(at'_n)}{(\bigwedge_{i=1}^n \varphi_i(\exists \vec{z}(prem\sigma \wedge \bigwedge_{x \in dom(\sigma)} x \equiv x\sigma)))} \Uparrow$$

where for all $1 \leq i \leq n$, $at'_i\sigma = at_i\sigma$ and φ_i does not contain existential quantifiers or negation or implication symbols.

$$\textit{application of (7)} \quad \frac{\varphi_1(at'_1) \vee \cdots \vee \varphi_n(at'_n)}{(\bigwedge_{i=1}^n \varphi_i(\exists \vec{z}(prem\sigma \wedge \bigwedge_{x \in dom(\sigma)} x \equiv x\sigma)))} \Uparrow$$

where for all $1 \leq i \leq n$, $at'_i\sigma = at_i\sigma$ and φ_i does not contain universal quantifiers or negation or implication symbols.

$$\textit{application of (11)} \quad \frac{\varphi_1(at'_1) \wedge \cdots \wedge \varphi_n(at'_n)}{(\bigvee_{i=1}^n \varphi_i(\forall \vec{z}(\bigwedge_{x \in dom(\sigma)} x \equiv x\sigma \Rightarrow conc\sigma)))} \Downarrow$$

where for all $1 \leq i \leq n$, $at'_i\sigma = at_i\sigma$ and φ_i does not contain existential quantifiers or negation or implication symbols.

$$\textit{application of (12)} \quad \frac{\varphi_1(at'_1) \vee \cdots \vee \varphi_n(at'_n)}{(\bigvee_{i=1}^n \varphi_i(\forall \vec{z}(\bigwedge_{x \in dom(\sigma)} x \equiv x\sigma \Rightarrow conc\sigma)))} \Downarrow$$

where for all $1 \leq i \leq n$, $at'_i\sigma = at_i\sigma$ and φ_i does not contain universal quantifiers or negation or implication symbols.

- **Tautology introduction.** Let $True \Longrightarrow conc$ and $False \Longleftarrow prem$ be valid in the specification's initial/final model.

$$\text{\textit{application of (13)}}\quad \frac{\varphi}{\forall\vec{z}conc \Rightarrow \varphi}\Updownarrow$$

$$\text{\textit{application of (14)}}\quad \frac{\varphi}{\neg\varphi \Rightarrow \exists\vec{z}prem}\Updownarrow$$

- **Strong coinduction.** Select subformulas of the form A and turn them into A'. Each axiom

$$p(\vec{t}) \quad \Longrightarrow \quad conc$$

is transformed into

$$p(\vec{t}) \quad \Longrightarrow \quad conc[p'/p] \vee p(\vec{t}).$$

A' is applied to the transformed axioms for p. The conjunction of the resulting clauses replaces A. p' is added as a new predicate to the current signature and

$$\begin{aligned} p'(\vec{x}) \quad &\Longleftarrow \quad \{prem_1 \wedge\}\ \vec{x} = \vec{t_1} \\ &\qquad \wedge \quad \ldots \\ &\qquad \wedge \quad \{prem_k \wedge\}\ \vec{x} = \vec{t_k}, \\ p'(\vec{x}) \quad &\Longleftarrow \quad p(\vec{x}) \end{aligned}$$

become the axioms of p'.

- **Strong fixpoint induction.** Select subformulas of the form B/C/D and turn them into B'/C'/D'. Each axiom

$$p(\vec{t}) \quad \Longleftarrow \quad prem$$

is transformed into

$$p(\vec{t}) \quad \Longleftarrow \quad prem[p'/p] \wedge p(\vec{t}).$$

Each axiom

$$f(\vec{t}) = t \quad \Longleftarrow \quad prem$$

is transformed into

$$f(\vec{t}) = t \quad \Longleftarrow \quad prem[f'(\vec{u}, u)/f(\vec{u}) = u] \wedge f(\vec{t}) = t.$$

B'/C'/D' is applied to the transformed (and flattened) axioms for f resp. p. The conjunction of the resulting clauses replaces B/C/D. p' resp. f' is added as a new predicate to the current signature and

$$\begin{array}{rcl} p'(\vec{x}) & \Longrightarrow & (\vec{x} = \vec{t_1} \ \Rightarrow \ conc_1) \\ & \wedge & \ldots \\ & \wedge & (\vec{x} = \vec{t_k} \ \Rightarrow \ conc_k), \\ p'(\vec{x}) & \Longrightarrow & p(\vec{x}) \end{array}$$

resp.

$$
\begin{array}{rcl}
f'(\vec{x}, x) & \Longrightarrow & ((\vec{x} = \vec{t_1} \wedge x = t_1) \;\Rightarrow\; conc_1) \\
& \wedge & \dots \\
& \wedge & ((\vec{x} = \vec{t_k} \wedge x = t_k) \;\Rightarrow\; conc_k), \\
f'(\vec{x}, x) & \Longrightarrow & f(\vec{x}) = x
\end{array}
$$

resp.

$$
\begin{array}{rcl}
f'(\vec{x}, x) & \Longrightarrow & (\vec{x} = \vec{t_1} \;\Rightarrow\; (x = t_1\{\wedge\ conc_1\})) \\
& \wedge & \dots \\
& \wedge & (\vec{x} = \vec{t_k} \;\Rightarrow\; (x = t_k\{\wedge\ conc_k\})), \\
f'(\vec{x}, x) & \Longrightarrow & f(\vec{x}) = x
\end{array}
$$

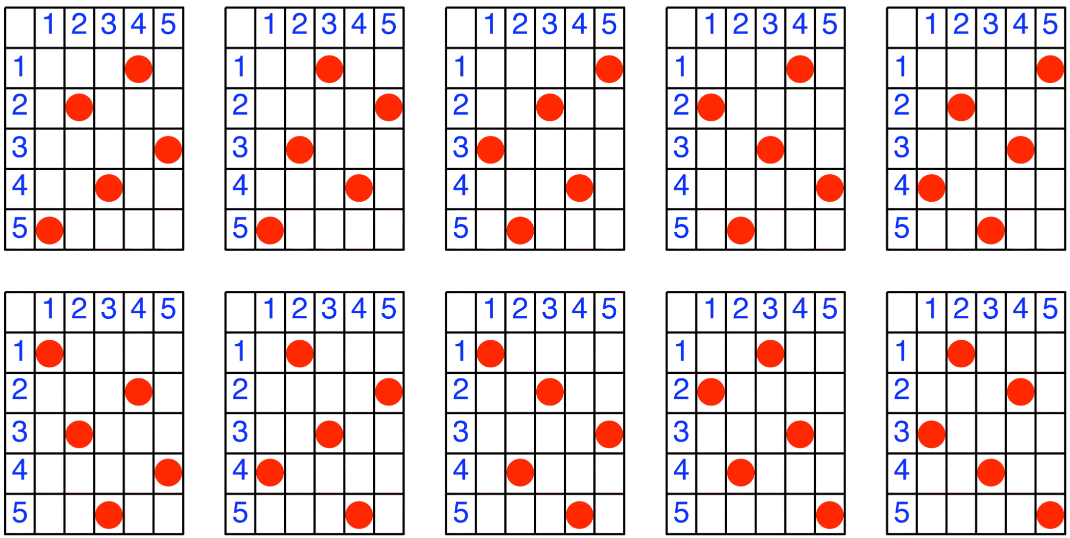
become the axioms for p' resp. f'.

More examples

Example Five queens (QUEENS)

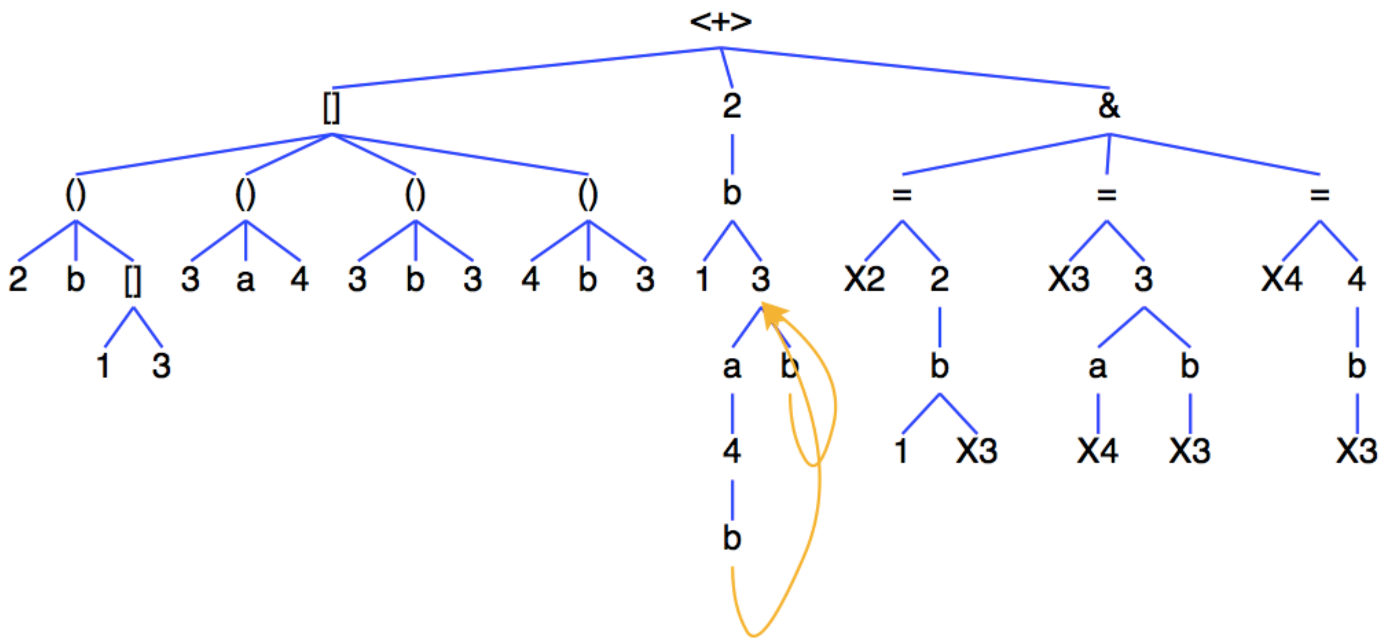
```
preds:     cmp loop queens
fovars:    n x y xs ys ps s s'

axioms:    (x =/= y+n & x =/= y-n ==> cmp(x)(y,n))              &
           (xs `gives` x & zipAll(cmp(x))(ys)[1..length(ys)]
                ==> (xs,ys) -> (xs-x,x:ys))                     &
           loop(xs,([],ys),zip(xs)(ys))                         &
           (s -> s' ==> (loop(xs,s,ps) <=== loop(xs,s',ps)))    &
           (xs = [1..n] ==> (queens(n,ps) <=== loop(xs,(xs,[]),ps)))
```

```
conjects:  queens(5,ps)
      ps = [(1,4),(2,2),(3,5),(4,3),(5,1)]
    | ps = [(1,3),(2,5),(3,2),(4,4),(5,1)]
    | ps = [(1,5),(2,3),(3,1),(4,4),(5,2)]
    | ps = [(1,4),(2,1),(3,3),(4,5),(5,2)]
    | ps = [(1,5),(2,2),(3,4),(4,1),(5,3)]
    | ps = [(1,1),(2,4),(3,2),(4,5),(5,3)]
    | ps = [(1,2),(2,5),(3,3),(4,1),(5,4)]
    | ps = [(1,1),(2,3),(3,5),(4,2),(5,4)]
    | ps = [(1,3),(2,1),(3,4),(4,2),(5,5)]
    | ps = [(1,2),(2,4),(3,1),(4,3),(5,5)]
```



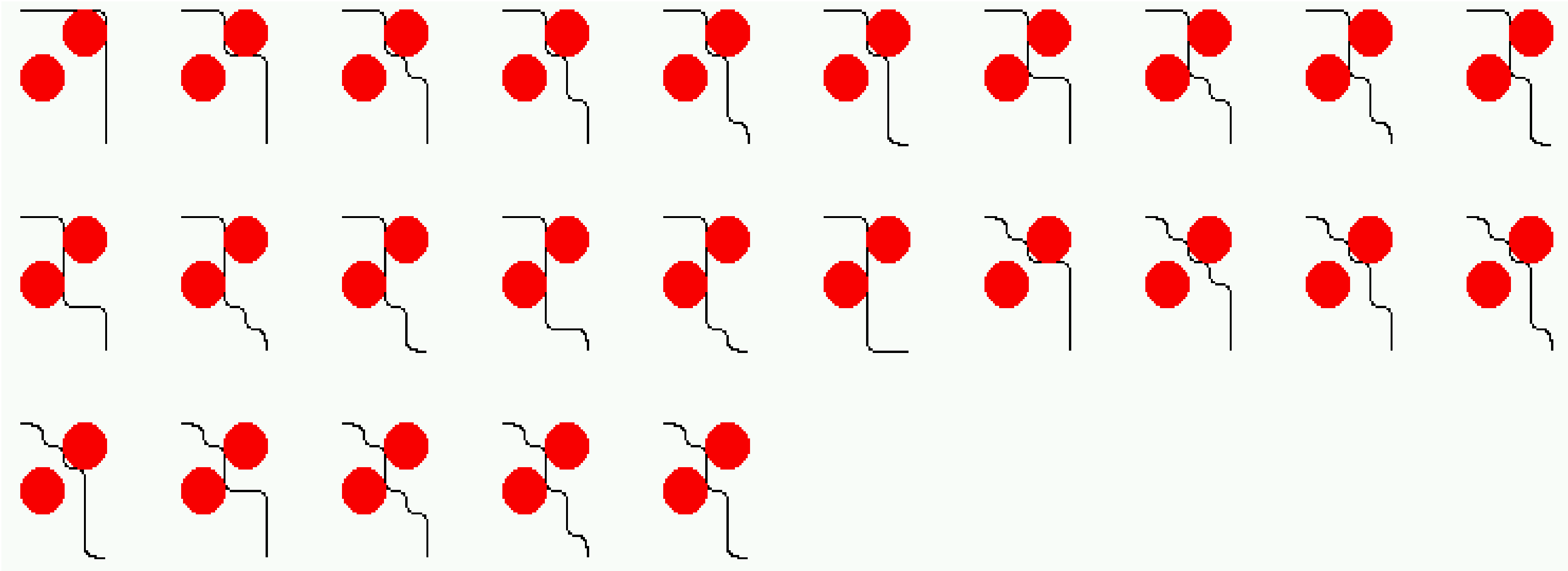
Example Labelled transition relation (TRANS1)

```
constructs: a b
defuncts:   states
axioms:     states = [1,2,3,4] & labels = [a,b] &
            (2,b) -> [1,3] & (3,b) -> 3 & (3,a) -> 4 & (4,b) -> 3
```



	a	b
2		1 3
3	4	3
4		3

Example Pathfinder (ROBOT)



```
preds:        loop
constructs:   turt path place circ red
defuncts:     cs
fovars:       x' y' p q ps pa
```

```
axioms:

cs = [(2,6),(6,2)]                                                   &

((p = (x+2,y) | p = (x,y+2)) & p `NOTin` cs ==> (x,y) -> p)          &

loop((8,12),path(ps),path(ps++[(8,12)]))                             &
(p < (8,12) & p -> q
     ==> (loop(p,path(ps),pa) <=== loop(q,path(ps++[p]),pa)))        &

((x,y) < (x,y') <=== y < y')                         &
((x,y) < (x',y) <=== x < x')                         &
((x,y) < (x',y') <=== x < x' & y < y')

conjects:

Any pa: (loop((0,0),path[],pa) & turt(pa:place(circ(2,red),cs)) = z)
```

Example Plan formation (ROBOTACTS)

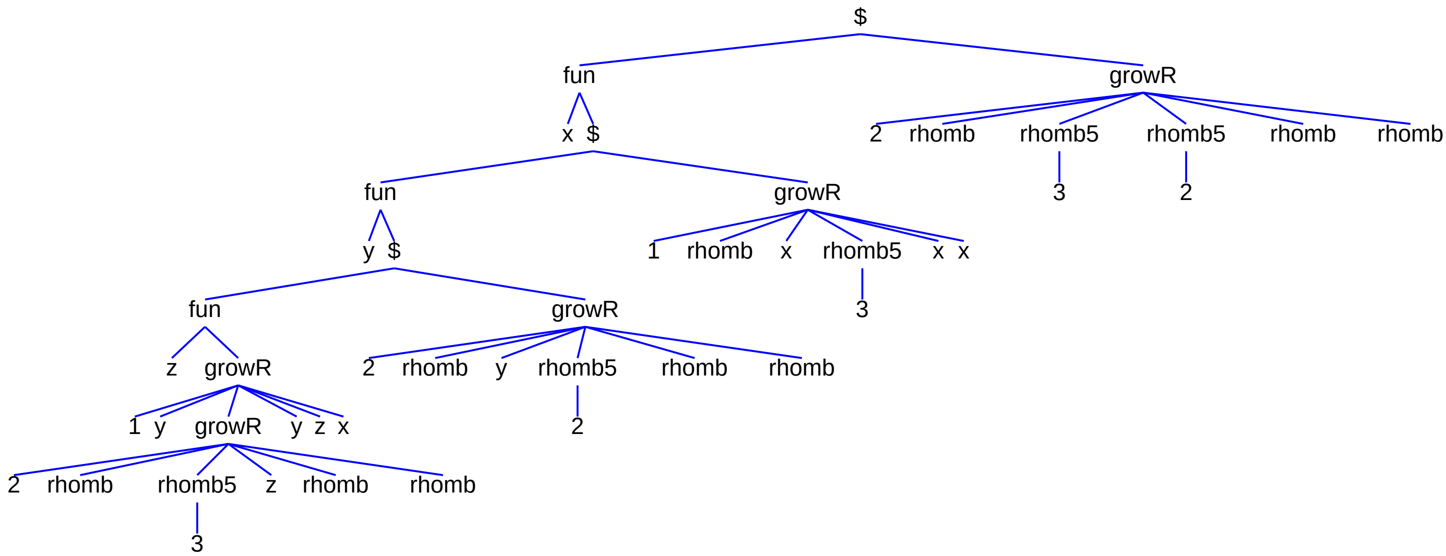
```
preds:      loop
constructs: turt pathS place circ red O C blue M R L
defuncts:   cs
fovars:     x' y' s s' act act' acts acts1 acts2

axioms:

(s = (x+2,y) & s `NOTin` cs ==> (x,y) -> (s,[M(2)]))            &
(s = (x,y+2) & s `NOTin` cs ==> (x,y) -> (s,[R,M(2),L]))        &

loop((8,12),acts,acts)                                           &
(s < (8,12) & s -> (s',acts)
   ==> (loop(s,acts1,acts2) <=== loop(s',acts1++acts,acts2)))   &

conjects:

Any acts: (loop((0,0),[],acts) &
           turt(O(blue):acts++[C,place(circ(2,red),cs)]) = z)
```

Example Pythagorean trees (PYTREE)

```
fovars:     x y
axioms:     trunk -> flipV(trunk)                &
            trunk -> grow(trunk,trunk)           &
            flipV(flipV(x)) -> x
conjects:   trunk                                <+>
            flipV(trunk)                         <+>
            grow(trunk,trunk)                    <+>
            grow(trunk,flipV(trunk))             <+>
            pytree1                              <+>
            pytree2                              <+>
            file(pytree1code)
```

Example Various trees (NICETREE)

```
fovars:     n x
axioms:     rhomb -> leaf(1.5,20)                                 &
            rhomb -> leafF(15,6)                                  &
            rhomb -> turt(blosF(10,5,2,red),blosF(5,3,1,yellow)) &
            rhomb -> polyR(5,[9,3])                               &
            rhomb -> rhomb5(1)                                    &
            rhomb -> flipV(rhomb)                                 &
            rhomb -> grow5(1,rhomb,rhomb,rhomb,rhomb,rhomb)       &
            rhomb -> growR(1,rhomb,rhomb,rhomb,rhomb,rhomb)       &
            ...                                                   &
            flipV(flipV(x)) -> x
```


$
fun
growR
x $
2 rhomb rhomb5 rhomb5 rhomb rhomb
3
2
fun
growR
y $
1 rhomb x rhomb5 x x
3
fun
growR
z growR
2 rhomb y rhomb5 rhomb rhomb
2
1 y growR y z x
2 rhomb rhomb5 z rhomb rhomb
3