Expander2:
program verification between interaction and automation

Peter Padawitz
TU Dortmund

May 27, 2018
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Components of Expander2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Types and functions</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Relations, quotients and substructures</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>3 specifications</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Deduction at a glance</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>Duality of narrowing and co/induction</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>Deduction in more detail</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>3 kinds of rules</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>The bottom level: Simplifications</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>Subsumption</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>The medium level: narrowing and rewriting</td>
<td>29</td>
</tr>
<tr>
<td>12</td>
<td>The top level: induction and coinduction</td>
<td>35</td>
</tr>
<tr>
<td>13</td>
<td>3 proofs</td>
<td>49</td>
</tr>
<tr>
<td>14</td>
<td>Ongoing work</td>
<td>53</td>
</tr>
</tbody>
</table>
15. The top level: More expansions

16. More examples
Components of Expander2

- **Enumerations of Formulas**
  - **Sums of Terms**
  - Creates and transforms
  - Applies

- **Inference Rules**
  - Axioms → Use
  - Applies
  - Applies
  - Infer

- **Derivations**
  - Performs
  - Performs

- **Solver**
  - Executed by
  - Displays
  - Stores

- **Reducer**
  - Computes
  - Evaluates

- **Simplifier**
  - Executes

- **Painter**
  - Draws

- **Haskell Functions**
  - Stepwise partial evaluation

- **Widgets Graphs**
  - Paths, polygons, tables, matrices, turtle systems

- **Rooted Graphs**
  - Alignments, partitions

- **Proof Terms**
3 representations of a formula/term: textual, tree-like (tree with additional forward or backward edges) 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 \rightarrow [n,n+1]) &
          (n < 6 & n \mod\ 2 =/= 0 ==> n \rightarrow n+1) &
          6 \rightarrow [1,3,5,7..10]
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\&

\[
\begin{align*}
&= & & = & & = & & = & & = \\
\text{0} & & \text{0} & & \text{1} & & \text{1} & & \text{2} & & \text{2} & & \text{3} & & \text{3} & & \text{4} & & \text{4} & & \text{5} & & \text{5} & & \text{6} & & \text{6} \\
\text{0} & & \text{1} & & \text{2} & & \text{2} & & \text{3} & & \text{4} & & \text{4} & & \text{5} & & \text{6} & & \text{1} & & \text{3} & & \text{5} & & \text{7} & & \text{8} & & \text{9} & & \text{10} & & \text{4} & & \text{2}
\end{align*}
\]
Types and functions

• **Sums** $\prod_{i \in I} t_i$ 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

<table>
<thead>
<tr>
<th>Constructors</th>
<th>Destructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{polyType}(T) \xrightarrow{c} T$</td>
<td>$T \xrightarrow{d} \text{polyType}(T)$</td>
</tr>
<tr>
<td>represent</td>
<td>represent</td>
</tr>
<tr>
<td>data generated by, context-free languages, “finite”, “inductive” data</td>
<td>transition systems, “non-deterministic” functions, streams and other “infinite” data</td>
</tr>
<tr>
<td>initial $T$-algebra</td>
<td>final $T$-coalgebra</td>
</tr>
</tbody>
</table>

• Functions $T \xrightarrow{f} \text{polyType}(T)$ are defined as initial catamorphisms (**recursion**).

• Functions $\text{polyType}(T) \xrightarrow{g} T$ are defined by as final anamorphisms (**corecursion**).
An element of an initial algebra and a final coalgebra, respectively
Relations, quotients and substructures

- **Horn clauses** $r(t) \iff \varphi$ define a **predicate** (least relation)  
  (safety predicate, transition system)  
as the least solution of $r(t) \iff \varphi$ 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]]) & 
          (\text{part}(x:y:s,[x]:p) \iff \text{part}(y:s,p)) & 
          (\text{part}(x:y:s,(x:s’):p) \iff \text{part}(y:s,s’:p)) & 
          \text{flatten}[] = [] & 
          \text{flatten}(s:p) = s++\text{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.
  
  \[
  \begin{align*}
  \text{prove } \varphi : & \quad \varphi \vdash \text{True} \\
  \text{solve } \varphi : & \quad \varphi \vdash r(c) \\
  \text{refute } \varphi : & \quad \neg \varphi \vdash \text{True} \\
  \text{verify } p : & \quad p(x) \Rightarrow \varphi \vdash \text{True} \\
  \text{evaluate } p : & \quad p(x) \Leftrightarrow \varphi \vdash \text{True} \\
  \text{evaluate } t : & \quad t \equiv x \vdash c \equiv x
  \end{align*}
  \]

- **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

\[ \phi(x) \Rightarrow p(x) \]
\[ \phi(x) \Rightarrow q(x) \]
\[ p(x) \Rightarrow \phi(x) \]
\[ q(x) \Rightarrow \phi(x) \]

True / False / solved formula

axioms(p)[\phi/p]

axioms(q)[\phi/q]

proof

evaluation

predicate

p(x) ==> \phi(x)

fixpoint induction

narrowing

narrowing

narrowing

narrowing

narrowing

narrowing

narrowing

narrowing

copredicate

q(x) ==> \phi(x)

copredicate

copredicate

proof

evaluation

coinduction
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 $\varphi_1$ like

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

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

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

Deterministic or non-deterministic rewritings of $t_1$:

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

The term () denotes \textit{undefined}. 
prove $\varphi_1$: \[ \varphi_n = \text{True} \]

refute $\varphi_1$: \[ \varphi_n = \text{False} \]

solve $\varphi_1$: \[ \varphi_n \text{ has the form} \]
\[ \exists Z_1 : x_1 = c_1 \land \cdots \land \exists Z_k : x_k = c_k \land \]
\[ \forall Z_{k+1} : x_{k+1} \neq c_{k+1} \land \cdots \land \forall Z_n : x_n \neq c_n, \]

$x_1, \ldots, x_n$ are different free variables,
$c_1, \ldots, c_n$ consist of constructors,
\[ \{(i, j) \mid c_i \text{ contains } x_j \}^+ \text{ is acyclic}. \]

evaluate $t$:
\[ \varphi_1 = (t = x), \varphi_n = (x = c) \]

rewrite $t$ logically:
\[ \varphi_1 = (t \to x), \varphi_n = (x = c_1) \land \cdots \land (x = c_k) \]
3 levels of automation ~ 3 kinds of rules

- A simplification $\frac{\varphi}{\psi} \Downarrow$ is applicable to any $C[\varphi]$: $C[\varphi] \Downarrow$

- An expansion $\frac{\varphi}{\psi} \Uparrow$ is applicable to $C[\varphi]$ if $\varphi$ is a positive subformula of $C[\varphi]$:
  
  \[
  \text{polarity(position}(\varphi), C[\varphi]) = true \implies \frac{C[\varphi]}{C[\psi]} \Uparrow
  \]

- A contraction $\frac{\varphi}{\psi} \Downarrow$ is applicable to $C[\varphi]$ if $\varphi$ is a negative subformula of $C[\varphi]$:
  
  \[
  \text{polarity(position}(\varphi), C[\varphi]) = false \implies \frac{C[\varphi]}{C[\psi]} \Uparrow
  \]
\[ \text{polarity}(w, \psi) = f(w, \psi, \text{true}) \]
\[ f(0w, \varphi \Rightarrow \psi, b) = f(w, \varphi, \text{not}(b)) \]
\[ f(1w, \varphi \Rightarrow \psi, b) = f(w, \psi, b) \]
\[ f(0w, \varphi \lor \psi, b) = f(w, \varphi, b) \]
\[ f(1w, \varphi \lor \psi, b) = f(w, \psi, b) \]
\[ f(0w, \forall x \varphi, b) = f(w, \varphi, b) \]
\[ f(\varepsilon, \varphi, b) = b \]

- 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**

  \[
  \begin{array}{cccccc}
  \varphi \land \text{True}  & \varphi \lor \text{False}  & \varphi \land \text{False}  & \varphi \lor \text{True}  & () \to t  & t <+> () \\
  \varphi  & \varphi  & \text{False}  & \text{True}  & \text{False}  & t \\
  \end{array}
  \]

- **Flattening**

  \[
  \begin{array}{cccc}
  \varphi \land (\psi_1 \land \cdots \land \psi_n)  & \varphi \lor (\psi_1 \lor \cdots \lor \psi_n) \\
  \varphi \land \psi_1 \land \cdots \land \psi_n  & \varphi \lor \psi_1 \lor \cdots \lor \psi_n \\
  \end{array}
  \]

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

  \[
  f(\ldots, t_1 <+> \ldots <+> t_n, \ldots) \\
  f(\ldots, t_1, \ldots) <+> \ldots <+> f(\ldots, t_n, \ldots) \\
  p(\ldots, t_1 <+> \ldots <+> t_n, \ldots) \\
  p(\ldots, t_1, \ldots) \lor \cdots \lor p(\ldots, t_n, \ldots) \\
  \varphi \land \forall \vec{x}(\psi_1 \lor \cdots \lor \psi_n) \\
  \forall \vec{x}((\varphi \land \psi_1) \lor \cdots \lor (\varphi \land \psi_n)) \quad \text{if no } x \in \vec{x} \text{ occurs freely } \varphi
  \]
• **Term decomposition** Let $c$ and $d$ be different constructors.

\[
\begin{align*}
\frac{c(t_1, \ldots, t_n) = c(u_1, \ldots, u_n)}{t_1 = u_1 \land \cdots \land t_n = u_n} & \quad \frac{c(t_1, \ldots, t_n) = d(u_1, \ldots, u_n)}{\text{False}} \\
\frac{c(t_1, \ldots, t_n) \neq c(u_1, \ldots, u_n)}{t_1 \neq u_1 \lor \cdots \lor t_n \neq u_n} & \quad \frac{c(t_1, \ldots, t_n) \neq d(u_1, \ldots, u_n)}{\text{True}}
\end{align*}
\]

• **Quantifier distribution**

\[
\begin{align*}
\forall \vec{x} (\varphi_1 \land \cdots \land \varphi_n) & \quad \exists \vec{x} (\varphi_1 \lor \cdots \lor \varphi_n) & \quad \exists \vec{x} (\varphi \Rightarrow \psi) \\
\forall \vec{x} \varphi_1 \land \cdots \land \forall \vec{x} \varphi_n & \quad \exists \vec{x} \varphi_1 \lor \cdots \lor \exists \vec{x} \varphi_n & \quad \forall \vec{x} \varphi \Rightarrow \exists \vec{x} \psi
\end{align*}
\]

\[
\begin{align*}
\exists \vec{x} (\varphi_1 \land \cdots \land \varphi_n) & \quad \forall \vec{x} (\varphi_1 \lor \cdots \lor \varphi_n) \\
\exists \vec{x}_1 \varphi_1 \land \cdots \land \exists \vec{x}_n \varphi_n & \quad \forall \vec{x}_1 \varphi_1 \lor \cdots \lor \forall \vec{x}_n \varphi_n
\end{align*}
\]

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 $\varphi_j$, $1 \leq j \leq n$, $j \neq i$.  

23
• **Removal of negation.** Negation symbols are moved to literal positions where they are replaced by complement predicates: $\neg P(t)$ is reduced to $\text{not}_P(t)$, $\neg \text{not}_P(t)$ is reduced to $P(t)$. Co-Horn/Horn axioms for $\text{not}_P$ can be generated automatically from Horn/Co-Horn axioms for $P$.

• **Removal of quantifiers.** Unused *bounded variables* are removed. *Successive quantifiers* are merged.
Subsumption

\[
\frac{\varphi \Rightarrow \psi}{\text{True}} \quad \frac{\varphi \Leftrightarrow \psi}{\psi \Rightarrow \varphi} \quad \frac{\varphi \Leftrightarrow \varphi}{\psi \Rightarrow \varphi} \quad \frac{\varphi \land (\psi \Rightarrow \theta)}{\varphi \land \theta} \quad \text{if } \varphi \text{ subsumes } \psi
\]

\[
\frac{\varphi_1 \lor \cdots \lor \varphi_n}{\varphi_1 \lor \cdots \lor \varphi_{n-1}} \quad \text{if } \varphi_1 \land \cdots \land \varphi_{n-1} \text{ subsumes } \varphi_n
\]

\[
\frac{\varphi_1 \lor \cdots \lor \varphi_n}{\varphi_1 \lor \cdots \lor \varphi_{n-1}} \quad \text{if } \varphi_n \text{ subsumes } \varphi_1 \lor \cdots \lor \varphi_{n-1}
\]

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 \( \theta \) denote a renaming of variables.
\[ \varphi \sim \psi \implies \varphi \text{ subsumes } \psi \]

\[ \varphi \text{ subsumes } \psi \implies \neg \psi \text{ subsumes } \neg \varphi \]

\[ \varphi' \text{ subsumes } \varphi \text{ and } \psi \text{ subsumes } \psi' \implies \varphi \Rightarrow \psi \text{ subsumes } \varphi' \Rightarrow \psi' \]

\[ \exists 1 \leq i \leq n : \varphi \text{ subsumes } \psi_i \implies \varphi \text{ subsumes } \psi_1 \lor \cdots \lor \psi_n \]

\[ \forall 1 \leq i \leq n : \varphi_i \text{ subsumes } \psi \implies \varphi_1 \lor \cdots \lor \varphi_n \text{ subsumes } \psi \]

\[ \forall 1 \leq i \leq n : \varphi \text{ subsumes } \psi_i \implies \varphi \text{ subsumes } \psi_1 \land \cdots \land \psi_n \]

\[ \exists 1 \leq i \leq n : \varphi_i \text{ subsumes } \psi \implies \varphi_1 \land \cdots \land \varphi_n \text{ subsumes } \psi \]

\[ \varphi(\vec{x}) \text{ subsumes } \psi(\vec{x}) \implies \exists \vec{x} \varphi(\vec{x}) \text{ subsumes } \exists \vec{y} \psi(\vec{y}) \]

\[ \varphi(\vec{x}) \text{ subsumes } \psi(\vec{x}) \implies \forall \vec{x} \varphi(\vec{x}) \text{ subsumes } \forall \vec{y} \psi(\vec{y}) \]

\[ \exists \theta, \vec{t} : \varphi \theta \sim \psi(\vec{t}) \implies \varphi \text{ subsumes } \exists \vec{x} \psi(\vec{x}) \]

\[ \exists \theta, \vec{t} : \varphi(\vec{t}) \sim \psi \theta \implies \forall \vec{x} \varphi(\vec{x}) \text{ subsumes } \psi \]

\[ \exists \theta, \vec{t}, \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\} : (\varphi_{i_1} \land \cdots \land \varphi_{i_k}) \theta \sim \psi(\vec{t}) \]

\[ \implies \varphi_1 \land \cdots \land \varphi_n \text{ subsumes } \exists \vec{x} \psi(\vec{x}) \]

\[ \exists \theta, \vec{t}, \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\} : \varphi(\vec{t}) \sim (\psi_{i_1} \lor \cdots \lor \psi_{i_k}) \theta \]

\[ \implies \forall \vec{x} \varphi(\vec{x}) \text{ subsumes } \psi_1 \lor \cdots \lor \psi_n \]
• **Elimination of equations and inequations.** Let $x \in \vec{x} \setminus var(t)$.

$$
\frac{\exists \vec{x}(x = t \land \varphi)}{\exists \vec{x}\varphi[t/x]}
\quad
\frac{\forall \vec{x}(x \neq t \lor \varphi)}{\forall \vec{x}\varphi[t/x]}
$$

$$
\frac{\forall \vec{x}(x = t \land \varphi \Rightarrow \psi)}{\forall \vec{x}(\varphi \Rightarrow \psi)[t/x]}
\quad
\frac{\forall \vec{x}(\varphi \Rightarrow x \neq t \lor \psi)}{\forall \vec{x}(\varphi \Rightarrow \psi)[t/x]}
$$

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

$$
\frac{\exists \vec{x}(x = t \land \varphi)}{\exists \vec{x}(x = t \land \varphi[t/x])}
\quad
\frac{\forall \vec{x}(x \neq t \lor \varphi)}{\forall \vec{x}(x \neq t \lor \varphi[t/x])}
$$

$$
\frac{\forall \vec{x}(x = t \land \varphi \Rightarrow \psi)}{\forall \vec{x}(x = t \land \varphi[t/x] \Rightarrow \psi[t/x])}
\quad
\frac{\forall \vec{x}(\varphi \Rightarrow x \neq t \lor \psi)}{\forall \vec{x}(\varphi[t/x] \Rightarrow x \neq t \lor \psi[t/x])}
$$
• **Universal quantification of implications**

\[
\begin{align*}
\exists \vec{x} \varphi & \Rightarrow \psi & \psi & \Rightarrow \forall \vec{x} \varphi \\
\forall \vec{x} (\varphi \Rightarrow \psi) & \quad \forall \vec{x} (\psi \Rightarrow \varphi)
\end{align*}
\]

if no variable of \( \vec{x} \) occurs freely in \( \psi \).

• **Implication splitting**

\[
\begin{align*}
\forall \vec{x} (\varphi_1 \lor \cdots \lor \varphi_n \Rightarrow \psi) & \\
\forall \vec{x} (\varphi_1 \Rightarrow \psi) \land \cdots \land \forall \vec{x} (\varphi_n \Rightarrow \psi) & \\
\forall \vec{x} (\varphi \Rightarrow \psi_1 \land \cdots \land \psi_n)
\end{align*}
\]

• **Uncurrying**

\[
\begin{align*}
\varphi \Rightarrow (\theta \Rightarrow \psi_1) \lor \psi_2 & \\
\varphi \land \theta & \Rightarrow \psi_1 \lor \psi_2
\end{align*}
\]
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 $at_1, \ldots, at_n$ be atoms.
\[\{\text{guard } \Rightarrow\} \ (f(\vec{t}) = u \ {\iff prem})\]

\[\{\text{guard } \Rightarrow\} \ (t_1 \wedge \ldots \wedge t_n \rightarrow u \ {\iff prem})\]

\[\{\text{guard } \Rightarrow\} \ (p(\vec{t}) \ {\iff prem})\]

\[\vec{t} = u \ {\iff prem}\]

\[q(\vec{t}) \ {\iff prem}\]

\[\text{at}_1 \wedge \ldots \wedge \text{at}_n \ {\iff prem}\]

\[\text{at}_1 \vee \ldots \vee \text{at}_n \ {\iff prem}\]

\[\{\text{guard } \Rightarrow\} \ (q(\vec{t}) \implies \text{conc})\]

\[\vec{t} = u \implies \text{conc}\]

\[p(\vec{t}) \implies \text{conc}\]

\[\text{at}_1 \wedge \ldots \wedge \text{at}_n \implies \text{conc}\]

\[\text{at}_1 \vee \ldots \vee \text{at}_n \implies \text{conc}\]

\[\text{conc}\]

\[\neg \text{prem}\]
narrowing upon a predicate $p$

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

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

(*) $\vec{x}$ is a list of the variables of $t$,

for all $1 \leq i \leq k$, $t \sigma_i = t_i \sigma_i$, $\gamma_i \sigma_i \vdash True$ and $Z_i = var(t_i, \varphi_i)$,

for all $k < i \leq n$, $t$ is not unifiable with $t_i$.

narrowing upon a copredicate $p$

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

where $\gamma_1 \Rightarrow (p(t_1) \implies \varphi_1), \ldots, \gamma_n \Rightarrow (p(t_n) \implies \varphi_n)$ are the axioms for $p$ and (*) holds true.
narrowing upon a defined function $f$

\[ r(\ldots, f(t), \ldots) \]

\[ \bigvee_{i=1}^{k} \exists Z_i : (r(\ldots, u_i, \ldots)\sigma_i \land \varphi_i \sigma_i \land \vec{x} = \vec{x}\sigma_i) \lor \]

\[ \bigvee_{i=k+1}^{l} (r(\ldots, f(t), \ldots)\sigma_i \land \vec{x} = \vec{x}\sigma_i) \]

where $r$ is a predicate or copredicate,

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

\((**\)\) $\vec{x}$ is a list of the variables of $t$,

- for all $1 \leq i \leq k$, $t\sigma_i = t_i\sigma_i$, $\gamma_i \sigma_i \vdash True$ and $Z_i = var(t_i, \varphi_i)$,
- for all $k < i \leq l$, $\sigma_i$ 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 $\to$

\[ t \land \upsilon \to t' \]

\[ \bigvee_{i=1}^{k} \exists Z_i : ((u_i \land \upsilon)\sigma_i = t'\sigma_i \land \varphi_i \sigma_i \land \vec{x} = \vec{x}\sigma_i) \lor \]

\[ \bigvee_{i=k+1}^{l} ((t \land \upsilon)\sigma_i \to t'\sigma_i \land \vec{x} = \vec{x}\sigma_i) \]

where $\gamma_1 \Rightarrow (t_1 \to u_1 \iff \varphi_1), \ldots, \gamma_n \Rightarrow (t_n \to u_n \iff \varphi_n)$ are the axioms for $\to$, (**\) holds true and $\sigma_i$ is a unifier modulo associativity and commutativity of $\land$
elimination of non-narrowable atoms and terms

\[
\begin{array}{cccc}
p(t) & q(t) & r(\ldots, f(t), \ldots) & t \rightarrow t'
\end{array}
\]

False \quad True \quad \quad r(\ldots, ()\ldots) \quad () \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 \( \gamma \Rightarrow (p(u) \iff \varphi), \gamma \Rightarrow (q(u) \implies \varphi), \gamma \Rightarrow (f(u) = v \iff \varphi) \) and \( \gamma \Rightarrow (u \rightarrow v \iff \varphi) \), \( t \) and \( u \) are not unifiable.
rewriting upon a defined function $f$

\[
\begin{align*}
\frac{c(f(t))}{c(u_1\sigma_1)<+> \cdots <+> c(u_k\sigma_k)}
\end{align*}
\]

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 $\gamma_i\sigma_i \vdash \text{True}$, for all $k < i \leq n$, $t$ does not match $t_i$.

rewriting upon the predicate $\rightarrow$

\[
\begin{align*}
\frac{c(t)}{c(u_1\sigma_1)<+> \cdots <+> c(u_k\sigma_k)}
\end{align*}
\]

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

\[
\begin{align*}
\frac{f(t)}{()}
\end{align*}
\]

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 \(\varphi = (\text{prem} \Rightarrow \text{conc})\), then the \textit{induction hypotheses}

\[
\text{conc}' \iff (x_1, \ldots, x_n) \gg (x'_1, \ldots, x'_n) \land \text{prem}'
\]

\[
\text{prem}' \Rightarrow ((x_1, \ldots, x_n) \gg (x'_1, \ldots, x'_n) \Rightarrow \text{conc}')
\]

are added to the current theorems.

If \(\varphi\) is not an implication, then

\[
\text{conc}' \iff (x_1, \ldots, x_n) \gg (x'_1, \ldots, x'_n)
\]

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 \(\varphi\) can be derived to \textit{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 \( p \)**

\[
\begin{align*}
\psi(x) \Rightarrow p(x) \\
\land_{p(t) \Rightarrow \varphi \in AX_p} (\psi(t) \Rightarrow \varphi[\psi/p])
\end{align*}
\]

**Realization:**

- Select subformulas

\[
\begin{align*}
\{\text{prem}_1 \Rightarrow\} & p(\vec{t}_1) \\
\land \ldots & \quad \quad \quad \quad \quad \quad \quad \quad \quad (A) \\
\land \{\text{prem}_k \Rightarrow\} & p(\vec{t}_k)
\end{align*}
\]

such that \( p \) does not depend on any predicate or function occurring in \( \text{prem}_i \).
(A) is turned into

\[ p(\vec{x}) \iff \{ \text{prem}_1 \land \} \vec{x} = \vec{t}_1 \]
\[ \land \ldots \]  
\[ \land \{ \text{prem}_k \land \} \vec{x} = \vec{t}_k \]

(A')

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

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

\[ p'(\vec{x}) \iff \{ \text{prem}_1 \land \} \vec{x} = \vec{t}_1 \]
\[ \land \ldots \]  
\[ \land \{ \text{prem}_k \land \} \vec{x} = \vec{t}_k \]

(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 $p$

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

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 $\varphi'$ be the result of submitting $\varphi$ 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 \( p \)

\[
\begin{align*}
p(x) & \Rightarrow \psi(x) \\
\land_{p(t) \Leftarrow \varphi \in AX_p}(\varphi[\psi/p] \Rightarrow \psi(t)) & \uparrow
\end{align*}
\]

**Realization:**

- Select subformulas

\[
p(t_1) \Rightarrow conc_1 \\
\land \ldots \\
\land p(t_k) \Rightarrow conc_k
\]

such that \( p \) does not depend on any predicate or function occurring in \( conc_i \).

(B) is turned into

\[
p(\vec{x}) \implies (\vec{x} = t_1 \Rightarrow conc_1) \\
\land \ldots \\
\land (\vec{x} = t_k \Rightarrow conc_k)
\]

(B')

where \( \vec{x} \) is a list of variables.
A new predicate \( p' \) is added to the current signature and

\[
p'(\vec{x}) \implies (\vec{x} = \vec{t}_1 \implies \text{conc}_1) \\
\quad \land \ldots \\
\quad \land (\vec{x} = \vec{t}_k \implies \text{conc}_k)
\]

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 $\varphi \vdash_{AX_p} \varphi'$.

**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 $\varphi'$ be the result of submitting $\varphi$ 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 $f$

\[
\begin{align*}
f(x) &\equiv y \Rightarrow \psi(x, y) \\
\land_{f(t)\equiv u\leftarrow\varphi\in\text{flat}(AX_f)}(\varphi[\psi/(f(-)\equiv -)] \Rightarrow \psi(t, u)) &\uparrow
\end{align*}
\]

Realization:

➢ Select subformulas

\[
f(t_1) = u_1 \Rightarrow \text{conc}_1
\]
\[
\land \ldots
\]
\[
\land f(t_k) = u_k \Rightarrow \text{conc}_k
\]

or

\[
f(t_1) = u_1 \{\land \text{conc}_1\}
\]
\[
\land \ldots
\]
\[
\land f(t_k) = u_k \{\land \text{conc}_k\}
\]

such that $f$ does not depend on any predicate or function occurring in $u_i$ or $\text{conc}_i$. 

42
(C) is turned into

\[ f(\vec{x}) = z \implies (\vec{x} = \vec{t}_1 \land z = u_1 \implies \text{conc}_1) \]
\[ \land \ldots \]
\[ \land (\vec{x} = \vec{t}_k \land z = u_k \implies \text{conc}_k), \]

(C’)

(D) is turned into

\[ f(\vec{x}) = z \implies (\vec{x} = \vec{t}_1 \implies z = u_1\{\land \text{conc}_1\}) \]
\[ \land \ldots \]
\[ \land (\vec{x} = \vec{t}_k \implies z = u_k\{\land \text{conc}_k\})\]

(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

$$f'(\vec{x}, z) \Rightarrow ((\vec{x} = \vec{t}_1 \land z = t_1) \Rightarrow \text{conc}_1)$$

$$\land \ldots$$

$$\land ((\vec{x} = \vec{t}_k \land z = t_k) \Rightarrow \text{conc}_k)$$

resp.

$$f'(\vec{x}, z) \Rightarrow (\vec{x} = \vec{t}_1 \Rightarrow (z = t_1 \{\land \text{conc}_1\}))$$

$$\land \ldots$$

$$\land (\vec{x} = \vec{t}_k \Rightarrow (z = t_k \{\land \text{conc}_k\}))$$

becomes the axiom for $f'$.

$AX_0$ is applied to $\text{flat}(AX_f)[f'/f(\_\) \equiv \_)]$.

The conjunction of the resulting clauses replaces the original conjecture C/D.
• \textit{n-level fixpoint induction on a function} \( f \)

\[
\begin{align*}
  f(x) & \equiv y \Rightarrow \psi(x, y) \\
  \bigwedge_{f(t) \equiv u \Leftarrow \varphi \in \text{flat}(AX_f)}(\varphi'[\psi/(f(\_ ) \equiv \_) \Rightarrow \psi(t, u)]) & \uparrow
\end{align*}
\]

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

\textit{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 \( f(t) \equiv u \Leftarrow \varphi \in \text{flat}(AX_f) \), let \( \varphi' \) be the result of submitting \( \varphi \) to a sequence of \( n \) inference steps each of which consists of the parallel application of \( \text{flat}(AX_f) \) 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

\[ f(t_1, \ldots, t_n) = t \Rightarrow \text{conc} \quad (A) \]

or

\[ f(t_1, \ldots, t_n) = t \{\land \text{conc}\} \quad (B) \]

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 \( \text{INV1} \land \text{INV2} \) where

\[ \text{INV}(x_1, \ldots, x_n, u_1, \ldots, u_k) \]

\[ g(x_i, \ldots, x_n, y_1, \ldots, y_k) = z \land \text{INV}(x_1, \ldots, x_n, y_1, \ldots, y_k) \]

\[ \Rightarrow (x_1 = t_1 \land \ldots \land x_n = t_n \land x = t \Rightarrow \text{conc}) \quad (\text{INV2}) \]

while \( B \) is turned into \( \text{INV1} \land \text{INV3} \) where

\[ g(x_i, \ldots, x_n, y_1, \ldots, y_k) = z \land \text{INV}(x_1, \ldots, x_n, y_1, \ldots, y_k) \]

\[ \Rightarrow (x_1 = t_1 \land \ldots \land x_n = t_n \Rightarrow x = t \{\land \text{conc}\}) \quad (\text{INV3}) \]
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 $INV_1 \land INV_2$ where

$$INV(x_1, \ldots, x_n, u_1, \ldots, u_k, z)$$

$$\Rightarrow (x_1 = t_1 \land \ldots \land x_n = t_n \land x = t \Rightarrow cong) \quad (INV1)$$

$$g(x_i, \ldots, x_n, y_1, \ldots, y_k) = z \Rightarrow INV(x_i, \ldots, x_n, y_1, \ldots, y_k, z) \quad (INV2)$$

while a selected conjecture of the form B is turned into $INV_1 \land INV_3$ where

$$g(x_i, \ldots, x_n, y_1, \ldots, y_k) = z \Rightarrow INV(x_i, \ldots, x_n, y_1, \ldots, y_k, z) \quad (INV3)$$

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

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

Applying fixpoint induction w.r.t.

\[
\text{part}([x],[[x]]) \\
& (\text{part}(x:y:s,[x]:p) \iff \text{part}(y:s,p)) \\
& (\text{part}(x:y:s,x:s':p) \iff \text{part}(y:s,s':p))
\]

classifies at position [] of the preceding formula leads to

\[
\text{All } x:([x] = \text{flatten}([[x]]) \& \\
\text{All } x \ y \ s \ p:(y:s = \text{flatten}(p) \implies x:y:s = \text{flatten}([x]:p)) \& \\
\text{All } x \ y \ s \ p \ s':(y:s = \text{flatten}(s':p) \implies x:y:s = \text{flatten}(x:s':p))
\]

The reducts have been simplified.

Applying the axiom resp. theorem
\( \text{flatten}(s:p) = s++\text{flatten}(p) \)

at positions \([2,0,1,1],[2,0,0,1],[1,0,1,1],[0,0,1]\) of the preceding formula leads to

\[ \boxed{} = \text{flatten}[\boxed{}] \]

The reducts have been simplified.

Narrowing the preceding formula leads to

True
Example  FAIRBLINK

\[ \text{fair(eq(0))(blink) \& fair(eq(0))(1:blink)} \]

Applying coinduction w.r.t.

\[ (\text{fair(f)(s) } \implies \text{ exists(f)(s) \& fair(f)(tail(s)))} \]

at position \[ [] \] of the preceding formula leads to

\[ \text{exists(eq(0))(1:blink) \& tail(blink) = 1:blink \& exists(eq(0))(blink) | exists(eq(0))(1:blink) \& tail(blink) = blink \& exists(eq(0))(blink)} \]

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 $\varphi$ and enter a formula $\psi$ into the solver’s entry field. If $\varphi$ has positive/negative polarity, then $\varphi$ is combined conjunctively/disjunctively with $\psi$. 
• **Vertical shift of quantifiers.** Select quantified arguments of a propositional operator \( op \), i.e. \( op \in \{ \land, \lor, \neg, \Rightarrow \} \). 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

\[
prem_1 \land \cdots \land prem_m \Rightarrow conc_1 \lor \cdots \lor conc_n,
\]

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

\[
prem_{i'_1} \land \cdots \land prem_{i'_r} \land \neg conc_{j'_1} \land \cdots \land \neg conc_{j'_l}
\]

\[
\Rightarrow \neg prem_{i_1} \lor \cdots \lor \neg prem_{i_k} \lor conc_{j'_1} \lor \cdots \lor conc_{j'_s}
\]

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

\[ \varphi = \exists \vec{x}(\varphi_1 \land \cdots \land \varphi_n) \]

with positive polarity or two summands of a disjunction

\[ \psi = \forall \vec{x}(\varphi_1 \lor \cdots \lor \varphi_n) \]

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 \( \phi \) whose parent node holds a conjunction or disjunction symbol. A copy of \( \phi \) 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:

\[ \&, |, =, = / =, \sim, \sim / \sim, +, *, ^, \{ \}. \]

The list of selected subtrees is reversed.
• Atom decomposition.

\[
\begin{align*}
\frac{f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n)}{t_1 = u_1 \land \cdots \land t_n = u_n} & \quad \begin{aligned}
\frac{f(t_1, \ldots, t_n) \neq f(u_1, \ldots, u_n)}{t_1 \neq u_1 \lor \cdots \lor t_n \neq u_n}
\end{aligned}
\end{align*}
\]

• Replacement by other sides.

\[
\begin{align*}
\frac{t = u \land \varphi(t)}{t = u \land \varphi(u)} & \quad \frac{t \neq u \lor \varphi(t)}{t \neq u \lor \varphi(u)}
\end{align*}
\]

\[
\begin{align*}
\frac{\varphi(t) \Rightarrow t \neq u \lor \psi(t)}{\varphi(u) \Rightarrow t \neq u \lor \psi(u)} & \quad \frac{t = u \land \varphi(t) \Rightarrow \psi(t)}{t = u \land \varphi(u) \Rightarrow \psi(u)}
\end{align*}
\]

• Transitivity. Select an atom \( tRt' \) with positive polarity or \( n - 1 \) factors

\[
t_1Rt_2, \ t_2Rt_3, \ \ldots, \ t_{n-1}Rt_n
\]

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 $\psi$ 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 $\psi$ has the form $tRu \iff \text{prem}$ where $R$ is symmetric and to the formula left/right of $\iff$ resp. $\implies$ in all other cases. The transformation is correct if the conclusion/premise of $\psi$ 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 $\text{prem}$ resp. $\text{conc}$ that do not occur in $at_1, \ldots, at_n$. 
application of (6) \[ \frac{\varphi_1(at'_1) \land \cdots \land \varphi_n(at'_n)}{(\land_{i=1}^n \varphi_i(\exists \vec{z}(\text{prem}\sigma \land \land_{x \in \text{dom}(\sigma)} x \equiv x\sigma)))} \uparrow \]

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

application of (7) \[ \frac{\varphi_1(at'_1) \lor \cdots \lor \varphi_n(at'_n)}{(\land_{i=1}^n \varphi_i(\exists \vec{z}(\text{prem}\sigma \land \land_{x \in \text{dom}(\sigma)} x \equiv x\sigma)))} \uparrow \]

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

application of (11) \[ \frac{\varphi_1(at'_1) \land \cdots \land \varphi_n(at'_n)}{(\lor_{i=1}^n \varphi_i(\forall \vec{z}(\land_{x \in \text{dom}(\sigma)} x \equiv x\sigma \Rightarrow \text{conc}\sigma)))} \downarrow \]

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

application of (12) \[ \frac{\varphi_1(at'_1) \lor \cdots \lor \varphi_n(at'_n)}{(\lor_{i=1}^n \varphi_i(\forall \vec{z}(\land_{x \in \text{dom}(\sigma)} x \equiv x\sigma \Rightarrow \text{conc}\sigma)))} \downarrow \]

where for all \(1 \leq i \leq n\), \(at'_i\sigma = at_i\sigma\) and \(\varphi_i\) does not contain universal quantifiers or negation or implication symbols.
• **Tautology introduction.** Let $True \Rightarrow conc$ and $False \Leftarrow prem$ be valid in the specification’s initial/final model.

\[
\begin{align*}
\text{application of (13)} & \quad \frac{\varphi}{\forall \vec{z}_{conc} \Rightarrow \varphi} \updownarrow \\
\text{application of (14)} & \quad \frac{\varphi}{\neg \varphi \Rightarrow \exists \vec{z}_{prem}} \updownarrow
\end{align*}
\]
• **Strong coinduction.** Select subformulas of the form \( A \) and turn them into \( A' \). Each axiom

\[
p(t) \implies conc
\]

is transformed into

\[
p(t) \implies conc[p'/p] \lor p(t).
\]

\( A' \) 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

\[
p'(\vec{x}) \iff \{ prem_1 \land \} \vec{x} = \vec{t}_1
\]

\[
\land \ldots
\]

\[
\land \{ prem_k \land \} \vec{x} = \vec{t}_k,
\]

\[
p'(\vec{x}) \iff p(\vec{x})
\]

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

$$p(\vec{t}) \iff prem$$

is transformed into

$$p(\vec{t}) \iff prem[p'/p] \land p(\vec{t}).$$

Each axiom

$$f(\vec{t}) = t \iff prem$$

is transformed into

$$f(\vec{t}) = t \iff prem[f'(\vec{u}, u)/f(\vec{u}) = u] \land f(\vec{t}) = t.$$ 

$B'/C'/D'$ 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

$$p'(\vec{x}) \implies (\vec{x} = \vec{t}_1 \implies conc_1)$$

$$\land \ldots$$

$$\land (\vec{x} = \vec{t}_k \implies conc_k),$$

$$p'(\vec{x}) \implies p(\vec{x})$$
resp.\[
f'(\vec{x}, x) \implies ((\vec{x} = \vec{t}_1 \land x = t_1) \implies \text{conc}_1)
\]
\[
\land \quad \ldots
\]
\[
\land ((\vec{x} = \vec{t}_k \land x = t_k) \implies \text{conc}_k),
\]
\[
f'(\vec{x}, x) \implies f(\vec{x}) = x
\]
resp.\[
f'(\vec{x}, x) \implies ((\vec{x} = \vec{t}_1 \implies (x = t_1 \{\land \text{conc}_1\})))
\]
\[
\land \quad \ldots
\]
\[
\land ((\vec{x} = \vec{t}_k \implies (x = t_k \{\land \text{conc}_k\}))),
\]
\[
f'(\vec{x}, x) \implies f(\vec{x}) = x
\]
become the axioms for $p'$ resp. $f'$. 
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)

\[
\begin{align*}
\text{ps} &= [(1,4), (2,2), (3,5), (4,3), (5,1)] \\
\text{ps} &= [(1,3), (2,5), (3,2), (4,4), (5,1)] \\
\text{ps} &= [(1,5), (2,3), (3,1), (4,4), (5,2)] \\
\text{ps} &= [(1,4), (2,1), (3,3), (4,5), (5,2)] \\
\text{ps} &= [(1,5), (2,2), (3,4), (4,1), (5,3)] \\
\text{ps} &= [(1,1), (2,4), (3,3), (4,5), (5,3)] \\
\text{ps} &= [(1,2), (2,5), (3,3), (4,4), (5,3)] \\
\text{ps} &= [(1,1), (2,3), (3,5), (4,2), (5,4)] \\
\text{ps} &= [(1,3), (2,1), (3,4), (4,2), (5,4)] \\
\text{ps} &= [(1,2), (2,4), (3,1), (4,3), (5,5)]
\end{align*}
\]
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
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 0 C blue M R L

**defuncts:** cs

**fovars:** x’ y’ s s’ act act’ acts acts1 acts2

**axioms:**

\[(s = (x+2,y) \& s \text{ ‘NOTin’ cs } \Rightarrow (x,y) \rightarrow (s, [M(2)])) \& (s = (x,y+2) \& s \text{ ‘NOTin’ cs } \Rightarrow (x,y) \rightarrow (s, [R,M(2),L])) \&

\text{loop}((8,12),acts,acts) \& (s < (8,12) \& s \rightarrow (s’,acts))

\Rightarrow (\text{loop}(s,acts1,acts2) \Leftarrow \text{loop}(s’,acts1++acts,acts2))) \&

**conjects:**

Any acts: (loop((0,0),[],acts) \&

\text{turt}(0(\text{blue}):acts++[C,place(circ(2,red),cs)]) = z)
Example Pythagorean trees (PYTREE)

fovrs: 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 \rightarrow \text{leaf}(1.5, 20) \quad \&
        rhomb \rightarrow \text{leafF}(15, 6) \quad \&
        rhomb \rightarrow \text{turt}(\text{blosF}(10, 5, 2, \text{red}), \text{blosF}(5, 3, 1, \text{yellow})) \quad \&
        rhomb \rightarrow \text{polyR}(5, [9, 3]) \quad \&
        rhomb \rightarrow \text{rhomb5}(1) \quad \&
        rhomb \rightarrow \text{flipV}(\text{rhomb}) \quad \&
        rhomb \rightarrow \text{grow5}(1, \text{rhomb}, \text{rhomb}, \text{rhomb}, \text{rhomb}, \text{rhomb}) \quad \&
        rhomb \rightarrow \text{growR}(1, \text{rhomb}, \text{rhomb}, \text{rhomb}, \text{rhomb}, \text{rhomb}, \text{rhomb}) \quad \&

\ldots

\text{flipV}(\text{flipV}(x)) \rightarrow x