Local Reasoning: Implementation

Localize - a hierarchical prover

Carsten Ihlemann

Max-Planck-Institut für Informatik

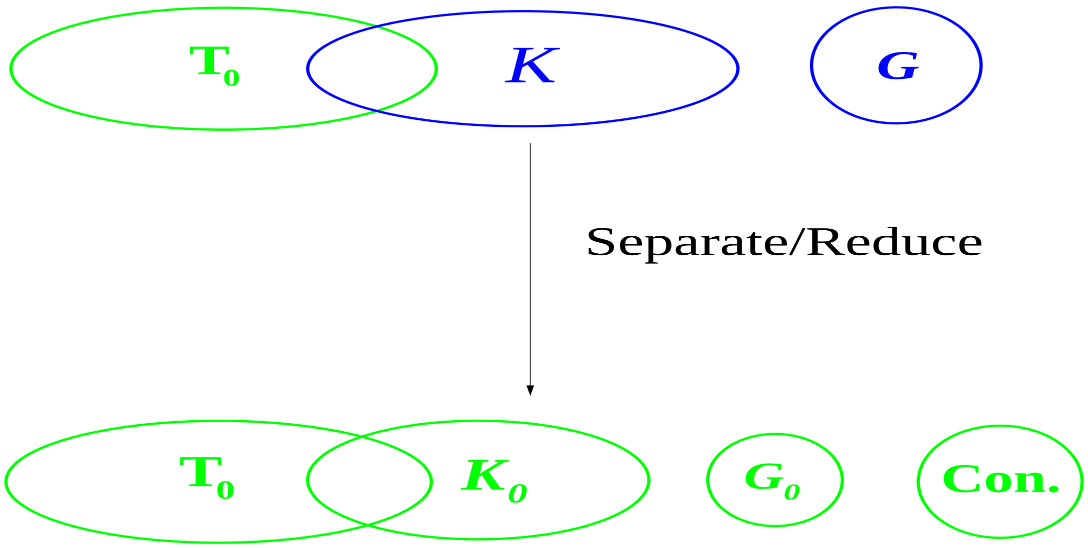
Locality Redux

- Task: Check entailment/satisfiability of formulas.
- Goal: Limit search space for counterexamples.
  $\Rightarrow$ Replace '$\forall$' by designated instances.
- Extending theory T_0 by (universal) clauses $\mathcal{K}$ and new functions.
- Designated instances $\mathcal{K}[G]$: all extension terms are ground terms in $\mathcal{K}$ or query G.

(Loc) $T_0 \cup \mathcal{K} \supseteq T_0$ is local, if for any set of ground clauses G:
$T_0 \cup \mathcal{K} \cup G \models \perp$ iff $T_0 \cup \mathcal{K}[G] \cup G$ has no (partial) model
(in which all ground terms in $\mathcal{K}$ and G are defined).

- Then: Reasoning in the extension becomes *reducible* to reasoning in the base theory.

Localize



- *Localize* implements reduction to base theory.
- *Localize* is a front end.

  $\Rightarrow$ integrated with SPASS, Yices, CVC and Redlog.
- *Localize* offers a full decision procedure ('sat' also).

Some local extensions

- Extensions by free/uninterpreted functions.
- Extensions by monotone functions.
- Extensions by functions defined by case distinctions.
- Fragments of the theory of arrays.

Examples

$(\mathsf{Mon_f}) \quad x \leq x', y \leq y' \rightarrow f(x, y) \leq f(x', y'),$

- The extension $T_0 \subseteq T_0 \cup (\mathsf{Mon_f})$ is local.
- Definitional extensions
  $\Rightarrow$ all extension functions are defined in base terms.
  $$\bigwedge_{i=1}^{k} \forall \bar{x}.(\phi_i(\bar{x}) \rightarrow f(\bar{x}) = t_i(\bar{x})), \qquad t_i \ \Sigma_0\text{-term} .$$
  with ϕ_i mutually exclusive (w.r.t T_0) Σ_0-clauses.
- All definitional extensions are local.

Bringing it all together

- We can split up large extensions.
- Example: inserting a new element into a sorted array a.
- The invariant is that the array remains sorted.

(1) $0 \leq i < n \wedge a[i] < x \rightarrow a'[i] = a[i]$

(2) $0 < i \leq n \wedge a[i-1] < x \leq a[i] \rightarrow a'[i] = x$

(3) $0 < i \leq n \wedge x \leq a[i] \rightarrow a'[i+1] = a[i]$

Bringing it all together

- We can split up large extensions.
- Example: inserting a new element into a sorted array a.
- The invariant is that the array remains sorted.

(1) $0 \leq i < n \wedge a[i] < x \rightarrow a'[i] = a[i]$

(2) $0 < i \leq n \wedge a[i-1] < x \leq a[i] \rightarrow a'[i] = x$

(3) $0 < i \leq n \wedge x \leq a[i] \rightarrow a'[i+1] = a[i]$

- Sortedness is monotonicity (local extension).

Bringing it all together

- We can split up large extensions.
- Example: inserting a new element into a sorted array a.
- The invariant is that the array remains sorted.

(1) $0 \leq i < n \wedge a[i] < x \rightarrow a'[i] = a[i]$

(2) $0 < i \leq n \wedge a[i-1] < x \leq a[i] \rightarrow a'[i] = x$

(3) $0 < i \leq n \wedge x \leq a[i] \rightarrow a'[i+1] = a[i]$

- Sortedness is monotonicity (local extension).
- The update of a to a' can be expressed as a definitional extension.

Bringing it all together

- We can split up large extensions.
- Example: inserting a new element into a sorted array a.
- The invariant is that the array remains sorted.

(1) $0 \leq i < n \wedge a[i] < x \rightarrow a'[i] = a[i]$

(2) $0 < i \leq n \wedge a[i-1] < x \leq a[i] \rightarrow a'[i] = x$

(3) $0 < i \leq n \wedge x \leq a[i] \rightarrow a'[i+1] = a[i]$

- Sortedness is monotonicity (local extension).
- The update of a to a’ can be expressed as a definitional extension.

$$T_0 \subseteq T_0 \cup \mathrm{Sorted}(\mathrm{a}, \mathrm{l}, \mathrm{u}) \subseteq \mathrm{T}_0 \cup \mathrm{Sorted}(\mathrm{a}, \mathrm{l}, \mathrm{u}) \cup \mathrm{Insert}(\mathrm{a}, \mathrm{l}, \mathrm{u}, \mathrm{x}, \mathrm{a}')$$

Example: Sorted Arrays [cont'd.]

Antecedents need to be mutually exclusive (for sorted a).

$$\begin{aligned}
x \leq a[0] \longrightarrow &\quad a'[0] = x \\
\forall i.\, x \leq a[0], 0 < i \leq n+1 \longrightarrow &\quad a'[i] = a[i-1]
\end{aligned}$$

$$\begin{aligned}
x \not\leq a[0], a[n] \leq x \longrightarrow &\quad a'[n+1] = x \\
\forall i.\, x \not\leq a[0], a[n] \leq x, -1 \leq i < n \longrightarrow &\quad a'[i+1] = a[i+1]
\end{aligned}$$

$$\begin{aligned}
\forall i.\, x < a[n], 0 \leq i < n, a[i] < x \leq a[i+1] \longrightarrow &\quad a'[i+1] = x \\
\forall i.\, a[0] < x < a[n], 0 \leq i < n, x \leq a[i], x \leq a[i+1] \longrightarrow &\quad a'[i+1] = a[i] \\
\forall i.\, a[0] < x < a[n], 0 \leq i < n, x \leq a[i], x \leq a[i+1] \longrightarrow &\quad a'[i] = a[i-1] \\
\forall i.\, a[0] < x < a[n], -1 \leq i < n, a[i+1] < x \longrightarrow &\quad a'[i+1] = a[i+1]
\end{aligned}$$

Example: Sorted Arrays [cont'd.]

Antecedents need to be mutually exclusive (for sorted a).

$$
\begin{aligned}
\forall i.\, i = 0, x \leq a[i] \rightarrow &\quad a'[i] = x \\
\forall i.\, x \leq a[0], 0 < i \leq n+1 \rightarrow &\quad a'[i] = a[i-1] \\[1em]
\forall i.\, i = n, x \not\leq a[0], a[i] \leq x \rightarrow &\quad a'[i+1] = x \\
\forall i.\, x \not\leq a[0], a[n] \leq x, -1 \leq i < n \rightarrow &\quad a'[i+1] = a[i+1] \\[1em]
\forall i.\, x < a[n], 0 \leq i < n, a[i] < x \leq a[i+1] \rightarrow &\quad a'[i+1] = x \\
\forall i.\, a[0] < x < a[n], 0 \leq i < n, x \leq a[i], x \leq a[i+1] \rightarrow &\quad a'[i+1] = a[i] \\
\forall i.\, a[0] < x < a[n], 0 \leq i < n, x \leq a[i], x \leq a[i+1] \rightarrow &\quad a'[i] = a[i-1] \\
\forall i.\, a[0] < x < a[n], -1 \leq i < n, a[i+1] < x \rightarrow &\quad a'[i+1] = a[i+1]
\end{aligned}
$$

Let's call these clauses $\mathsf{Update}(\mathsf{a}, \mathsf{a}')$.

Example (concluded)

- We wanted to show that
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}') \models \mathsf{Mon}(\mathsf{a}')$.
- First we negate and skolemize.
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}'), 0 \leq \mathsf{k} \leq \mathsf{l} \leq \mathsf{n}, \mathsf{a}'[\mathsf{k}] \not\leq \mathsf{a}'[\mathsf{l}] \models \perp$.

Example (concluded)

- We wanted to show that
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}') \models \mathsf{Mon}(\mathsf{a}')$.
- First we negate and skolemize.
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}'), \mathsf{G} \models \perp$.

Example (concluded)

- We wanted to show that
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}') \models \mathsf{Mon}(\mathsf{a}')$.
- First we negate and skolemize.
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}'), \mathsf{G} \models \perp$.
- $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}) \subseteq \mathsf{T}_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}')$ is a local extension (definitional).

Example (concluded)

- We wanted to show that
  $T_{\mathbb{Z}}, \mathsf{Mon}(a), \mathsf{Update}(a, a') \models \mathsf{Mon}(a')$.
- First we negate and skolemize.
  $T_{\mathbb{Z}}, \mathsf{Mon}(a), \mathsf{Update}(a, a'), G \models \perp$.
- $T_{\mathbb{Z}}, \mathsf{Mon}(a) \subseteq \mathrm{T}_{\mathbb{Z}}, \mathsf{Mon}(a), \mathsf{Update}(a, a')$ is a local extension (definitional).
- Hence, we can get rid of everything a' by reduction.
  $T_{\mathbb{Z}}, \mathsf{Mon}(a), \mathsf{Update}(a, a')_0, G_0, N_0 \models \perp$.

Example (concluded)

- We wanted to show that
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}') \models \mathsf{Mon}(\mathsf{a}')$.
- First we negate and skolemize.
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}'), \mathsf{G} \models \perp$.
- $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}) \subseteq \mathsf{T}_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}')$ is a local extension (definitional).
- Hence, we can get rid of everything a' by reduction.
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \underbrace{\mathsf{Update}(\mathsf{a}, \mathsf{a}')_0, \mathsf{G}_0, \mathsf{N}_0}_{\mathsf{G}_{\mathsf{new}}} \models \perp$.

Example (concluded)

- We wanted to show that
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}') \models \mathsf{Mon}(\mathsf{a}')$.
- First we negate and skolemize.
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}'), \mathsf{G} \models \perp$.
- $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}) \subseteq \mathsf{T}_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}')$ is a local extension (definitional).
- Hence, we can get rid of everything a' by reduction.
  $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}')_0, \mathsf{G}_0, \mathsf{N}_0 \models \perp$.
- $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a})$ is itself a local extension of $T_{\mathbb{Z}}$.

Example (concluded)

- We wanted to show that $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}') \models \mathsf{Mon}(\mathsf{a}')$.
- First we negate and skolemize. $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}'), \mathsf{G} \models \perp$.
- $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}) \subseteq \mathsf{T}_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}')$ is a local extension (definitional).
- Hence, we can get rid of everything a' by reduction. $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a}), \mathsf{Update}(\mathsf{a}, \mathsf{a}')_0, \mathsf{G}_0, \mathsf{N}_0 \models \perp$.
- $T_{\mathbb{Z}}, \mathsf{Mon}(\mathsf{a})$ is itself a local extension of $T_{\mathbb{Z}}$.
- In a second step we reduce everything to $T_{\mathbb{Z}}$ for which competent provers are abundant (e.g. Yices).
- Yices solves this problem in 0.2s. For the original problem Yices needs 7min.