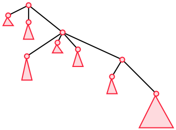
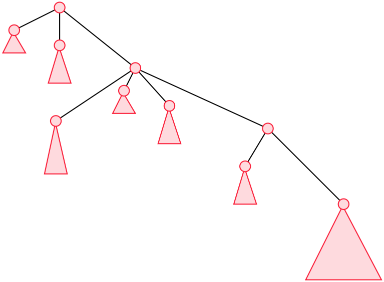
A Logic Your Typechecker Can Count On: Unordered Tree Types in Practice

Nate Foster (Penn)
Benjamin C. Pierce (Penn)
Alan Schmitt (INRIA Rhône-Alpes)



16 Feb 2007

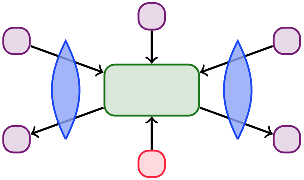
$$\mu X.\,\{\}\,|\,(hd\,[T] + tl\,[X])$$
$$\rightsquigarrow$$
$$\phi(x_0, .., x_4),\ \left[\begin{array}{l} \mathtt{hd}[T]\,, \mathtt{hd}[\neg T]\,, \\ \mathtt{tl}[X]\,, \mathtt{tl}[\neg X]\,, \\ \overline{\{\mathtt{hd}, \mathtt{tl}\}}[\mathsf{True}] \end{array}\right]$$

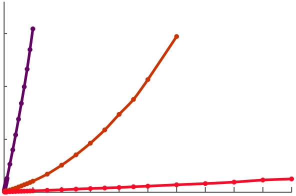


$$\mu X.\,\{\}\,|\,(hd\,[T]+tl\,[X])$$

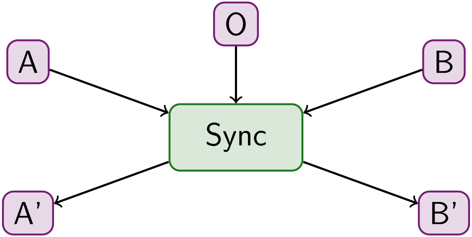
$$\Downarrow$$

$$\phi(x_0, .., x_4),\ \left[\begin{array}{l} \mathtt{hd}[T]\,,\mathtt{hd}[\neg T]\,, \\ \mathtt{tl}[X]\,,\mathtt{tl}[\neg X]\,, \\ \hline \{\mathtt{hd},\mathtt{tl}\}[\mathsf{True}] \end{array}\right]$$





Types in HARMONY

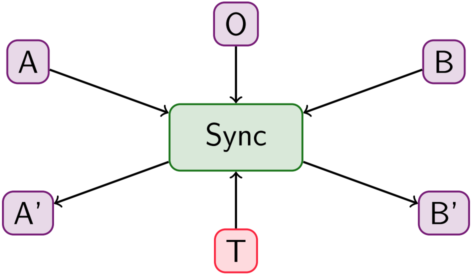


Harmony

A generic synchronization framework

- Architecture takes two replicas + original ⇒ updated replicas.
- Data model is “deterministic” trees: unordered, edge-labeled trees.

Types in HARMONY

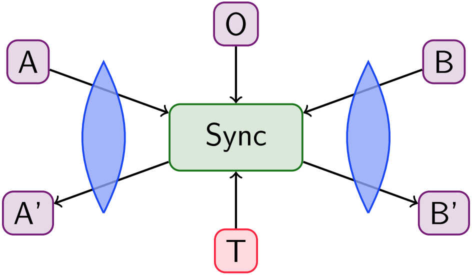


Harmony: Typed Synchronization [DBPL '05]

Behavior of synchronizer guided by type.

- If inputs well-typed, so are outputs.
- Required operations: membership of trees in type [also sets of names].

Types in HARMONY



Harmony: Lenses [POPL '05]

Pre-/post-process replicas using bi-directional programs.

- Facilitates heterogeneous synchronization.
- Types in conditionals, run-time asserts, static checkers.
- Required operations: membership, inclusion, equivalence, emptiness, [projection, injection, etc.].

Deterministic Tree Types

Syntax

$$
\begin{array}{rl}
T ::= & \texttt{\{\}} \mid n\texttt{[}T\texttt{]} \mid T\texttt{+}T \mid T\,\texttt{|}\,T \mid \texttt{\~{}}T \mid X \\
\mid & \texttt{!}\backslash\{n_1, .., n_k\}\texttt{[}T\texttt{]} \mid \texttt{*}\backslash\{n_1, .., n_k\}\texttt{[}T\texttt{]}
\end{array}
$$

Deterministic Tree Types

Syntax

$$T ::= \{\} \mid n[T] \mid T + T \mid T \mid T \mid \tilde{\ } T \mid X$$
$$\mid \; !\backslash\{n_1, .., n_k\}[T] \mid *\backslash\{n_1, .., n_k\}[T]$$

Semantics

Singleton denoting the unique tree with no children:

$$\circ \in \{\}$$

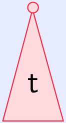
Deterministic Tree Types

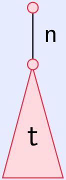
Syntax

$$
\begin{array}{rl}
T ::= & \{\} \mid n[T] \mid T+T \mid T \,|\, T \mid \,\tilde{}\, T \mid X \\
\mid & !\backslash\{n_1, .., n_k\}[T] \mid *\backslash\{n_1, .., n_k\}[T]
\end{array}
$$

Semantics

Atoms: trees with single child n and subtree in T:

If 

$\in T$, then 

$\in n[T]$

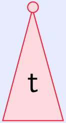
Deterministic Tree Types

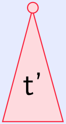
Syntax

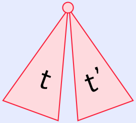
$$T ::= \{\} \mid n[T] \mid T{+}T \mid T \,|\, T \mid \sim T \mid X$$
$$\mid \; !\backslash\{n_1, .., n_k\}[T] \mid *\backslash\{n_1, .., n_k\}[T]$$

Semantics

Commutative concatenation operator:

If 

$\in T$ and 

$\in T'$, then 

$\in T{+}T'$

Deterministic Tree Types

Syntax

$$
\begin{array}{rl}
T ::= & \{\} \mid n[T] \mid T{+}T \mid T \,|\, T \mid \tilde{}\,T \mid X \\
\mid & !\backslash\{n_1, .., n_k\}[T] \mid *\backslash\{n_1, .., n_k\}[T]
\end{array}
$$

Semantics

Boolean operations and recursion:

$$
\begin{array}{rcl}
X_1 & = & T_1 \\
 & \vdots & \\
X_n & = & T_n
\end{array}
$$

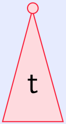
Deterministic Tree Types

Syntax

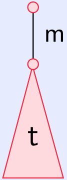
$$
\begin{array}{rl}
T ::= & \{\} \mid n\,[T] \mid T + T \mid T \,|\, T \mid \sim T \mid X \\
\mid & !\backslash\{n_1, .., n_k\}\,[T] \mid *\backslash\{n_1, .., n_k\}\,[T]
\end{array}
$$

Semantics

If $m \notin \{n_1, .., n_k\}$ and



$\in T$, then



$\in !\backslash\{n_1, .., n_k\}\,[T]$

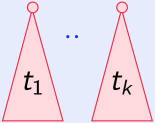
Deterministic Tree Types

Syntax

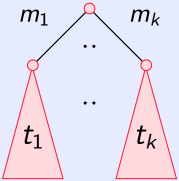
$$
\begin{array}{rl}
T ::= & \{\} \mid n[T] \mid T+T \mid T \mid T \mid \tilde{}\,T \mid X \\
\mid & !\backslash\{n_1, .., n_k\}[T] \mid *\backslash\{n_1, .., n_k\}[T]
\end{array}
$$

Semantics

If $m_1, .., m_k \notin \{n_1, .., n_k\}$ and



$\in T$, then



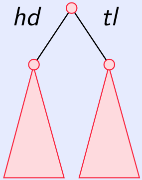
$\in *\backslash\{n_1, .., n_k\}[T]$

Deterministic Tree Types

Syntax

$$T ::= \{\} \mid n[T] \mid T{+}T \mid T \,|\, T \mid \tilde{\ } T \mid X$$
$$\mid \quad !\backslash\{n_1, .., n_k\}[T] \mid *\backslash\{n_1, .., n_k\}[T]$$

Example: *hd* [True]+*tl* [True]

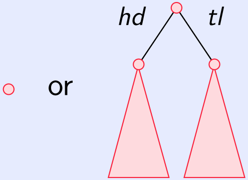


Deterministic Tree Types

Syntax

$$
\begin{array}{rl}
T ::= & \{\} \mid n[T] \mid T + T \mid T \,|\, T \mid \tilde{}\, T \mid X \\
\mid & !\backslash\{n_1, .., n_k\}[T] \mid *\backslash\{n_1, .., n_k\}[T]
\end{array}
$$

Example: $\{\} \mid (hd\,[\mathrm{True}] + tl\,[\mathrm{True}])$

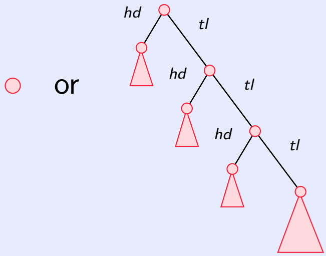


Deterministic Tree Types

Syntax

$$
\begin{array}{rl}
T ::= & \{\} \mid n\,[T] \mid T + T \mid T \mid T \mid \tilde{}\,T \mid X \\
\mid & !\backslash\{n_1, .., n_k\}\,[T] \mid *\backslash\{n_1, .., n_k\}\,[T]
\end{array}
$$

Example: $X = \{\} \mid (hd\,[\mathrm{True}] + tl\,[X])$

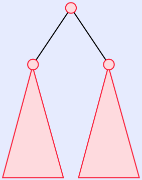


Deterministic Tree Types

Syntax

$$
\begin{array}{rll}
T ::= & \{\} \mid n[T] \mid T+T \mid T \,|\, T \mid \tilde{\ } T \mid X \\
\mid & !\backslash\{n_1,..,n_k\}[T] \mid *\backslash\{n_1,..,n_k\}[T]
\end{array}
$$

Example: ![True]+![True]

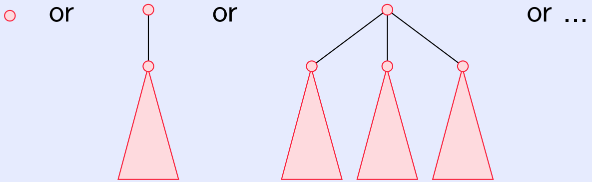


Deterministic Tree Types

Syntax

$$
\begin{array}{rl}
T ::= & \{\} \mid n[T] \mid T+T \mid T \,|\, T \mid \tilde{\ } T \mid X \\
\mid & !\backslash\{n_1, .., n_k\}[T] \mid *\backslash\{n_1, .., n_k\}[T]
\end{array}
$$

Example: ~(![True]+![True])



Can eliminate negations, and use direct algorithms, but types get large...

Sheaves Formulas

Formulas

$$S = \begin{array}{l} \phi(x_0, .., x_k), \\ [r_0[S_0]\,, .., r_k[S_k]] \end{array}$$

where ϕ is a Presburger formula and r_i a set of names.

[Dal Zilio, Lugiez, Meyssonnier, POPL '04]

Sheaves Formulas

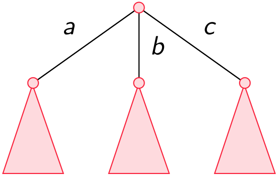
Formulas

$$S = \begin{array}{l} \phi(x_0, .., x_k), \\ [r_0[S_0], .., r_k[S_k]] \end{array}$$

where ϕ is a Presburger formula and r_i a set of names.

$\phi(x_0, x_1)$,
$[b[\text{True}], \{a, c\}[\text{True}]]$

0	0



Sheaves Formulas

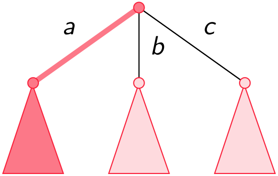
Formulas

$$S = \begin{array}{l} \phi(x_0, .., x_k), \\ [r_0[S_0]\,, .., r_k[S_k]] \end{array}$$

where ϕ is a Presburger formula and r_i a set of names.

$\phi(x_0, x_1)$,
$[b[\text{True}], \{a, c\}[\text{True}]]$

0	1



Sheaves Formulas

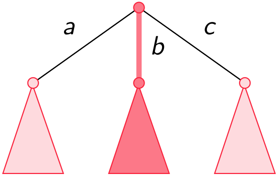
Formulas

$$S = \begin{array}{l}\phi(x_0, .., x_k),\\ \left[r_0[S_0]\,, .., r_k[S_k]\right]\end{array}$$

where ϕ is a Presburger formula and r_i a set of names.

$\phi(x_0, x_1)$,
$[b[\mathrm{True}], \{a, c\}[\mathrm{True}]]$

1	1



Sheaves Formulas

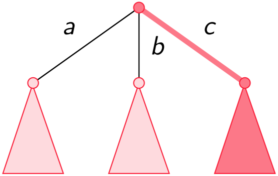
Formulas

$$S = \begin{array}{l} \phi(x_0, .., x_k), \\ [r_0[S_0]\,, .., r_k[S_k]] \end{array}$$

where ϕ is a Presburger formula and r_i a set of names.

$\phi(x_0, x_1),$
$[b[\text{True}], \{a, c\}[\text{True}]]$

1	2



Sheaves Formulas

Formulas

$$S = \begin{array}{l} \phi(x_0, .., x_k), \\ [r_0[S_0]\,, .., r_k[S_k]] \end{array}$$

where ϕ is a Presburger formula and r_i a set of names.

$\phi(x_0, x_1)$,
$[b[\text{True}], \{a, c\}[\text{True}]]$

1	2

$$\overset{?}{\models} \phi(1, 2)$$

Sheaves Formulas

Formulas

$$S = \begin{array}{l} \phi(x_0, .., x_k), \\ \left[r_0[S_0]\,, .., r_k[S_k]\right] \end{array}$$

where ϕ is a Presburger formula and r_i a set of names.

$$\begin{array}{l} \phi(x_0, x_1, x_2), \\ \left[b[\text{True}], \{a, c\}[\text{True}], \overline{\{a, b, c\}}[\text{True}] \right] \end{array}$$

For coherence: $r_i[S_i]$ must partition set of atoms.
Note: does not ensure determinism.

Examples as Sheaves Formulas

$X = (\{\}\,|\,\mathtt{hd}\,[\mathsf{True}]\,\mathtt{+tl}\,[X]\,)$

$$X = \begin{array}{l} (x_0 = x_1 = x_2 = x_3 = 0) \vee \\ (x_0 = x_1 = 1 \wedge x_2 = x_3 = 0), \\ \Big[\mathtt{hd}[\mathsf{True}]\,, \mathtt{tl}[X]\,, \mathtt{tl}[\neg X]\,, \overline{\{\mathtt{hd}, \mathtt{tl}\}}[\mathsf{True}]\Big] \end{array}$$

Examples as Sheaves Formulas

$X = (\{\}\,|\,\texttt{hd}\,[\text{True}]\,{+}\,\texttt{tl}\,[X]\,)$

$$X = \begin{array}{l} (x_0 = x_1 = x_2 = x_3 = 0) \vee \\ (x_0 = x_1 = 1 \wedge x_2 = x_3 = 0), \\ \left[\texttt{hd}[\text{True}]\,,\texttt{tl}[X]\,,\texttt{tl}[\neg X]\,,\overline{\{\texttt{hd},\texttt{tl}\}}[\text{True}]\right] \end{array}$$

~(![True]+![True])

$$x_0 \neq 2,$$
$$\left[\overline{\{\}}[\text{True}]\right]$$

Challenges and Strategies

Blowup in naive compilation from types to formulas.

- Syntactic optimizations avoid blowup in common cases.

Backtracking in top-down, non-deterministic traversal.

- Incremental algorithm avoids useless paths.

Presburger arithmetic requires double-exponential time.

- Compile Presburger formulas to MONA representation.
- Hash-consing allocation + aggressive memoization.

Challenges and Strategies

Blowup in naive compilation from types to formulas.

- Syntactic optimizations avoid blowup in common cases.

Backtracking in top-down, non-deterministic traversal.

- Incremental algorithm avoids useless paths.

Presburger arithmetic requires double-exponential time.

- Compile Presburger formulas to MONA representation.
- Hash-consing allocation + aggressive memoization.

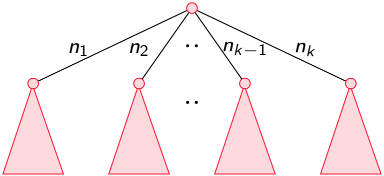
Contributions

- Strategies and algorithms;
- Implementation in Harmony;
- Experimental results.

Incremental Algorithm

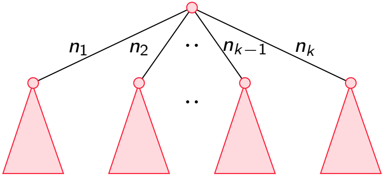
$\phi(x_0, .., x_k)$,
$[r_0[S_0], ..r_k[S_k]]$

0	0	..	0



Incremental Algorithm

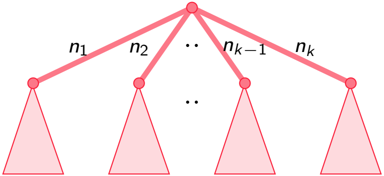
$\phi(x_0, .., x_k),$
$[r_0[S_0], ..r_k[S_k]]$ (ϕ)



Incremental Algorithm

$\phi(x_0, .., x_k)$,
$[r_0[S_0], ..r_k[S_k]]$

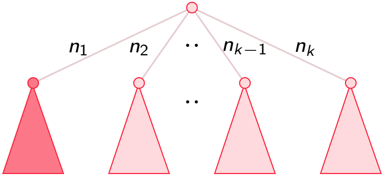
$(\phi \wedge \psi_{\mathsf{dom}})$



Incremental Algorithm

$\phi(x_0, .., x_k)$,
$[r_0[S_0], ..r_k[S_k]]$

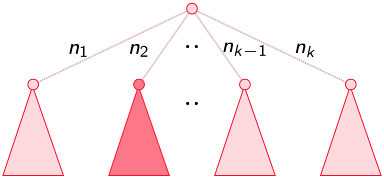
$(\phi \wedge \psi_{\mathsf{dom}} \wedge \psi_1)$



Incremental Algorithm

$\phi(x_0, .., x_k),$
$[r_0[S_0], ..r_k[S_k]]$

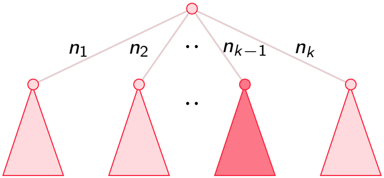
$(\phi \wedge \psi_{\text{dom}} \wedge \psi_1 \wedge \psi_2)$



Incremental Algorithm

$\phi(x_0, .., x_k)$,
$[r_0[S_0], ..r_k[S_k]]$

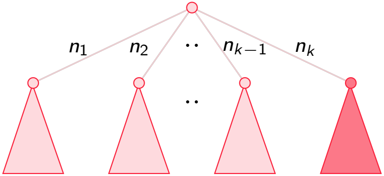
$(\phi \wedge \psi_{\text{dom}} \wedge \psi_1 \wedge .. \wedge \psi_{k-1})$



Incremental Algorithm

$\phi(x_0, .., x_k)$,
$[r_0[S_0], ..r_k[S_k]]$

$(\phi \wedge \psi_{\mathsf{dom}} \wedge \psi_1 \wedge .. \wedge \psi_k)$



Hash-Consing and Memoization

Thousands of formulas and trees, but many repeats.

Suggests hash-consed allocation:

- Sheaves formulas;
- Presburger formulas;
- Trees.

Memoization of intermediate results:

- MONA representations of Presburger formulas;
- Satisfiability of Presburger formulas;
- Membership results;
- Partially-evaluated member functions.

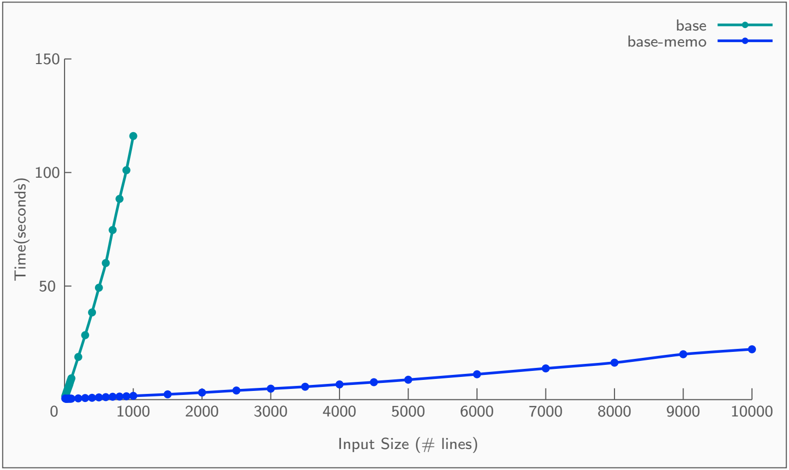
Experiments

Programs:

- Structured text parser;
- Address book validator;
- iCalendar lens.

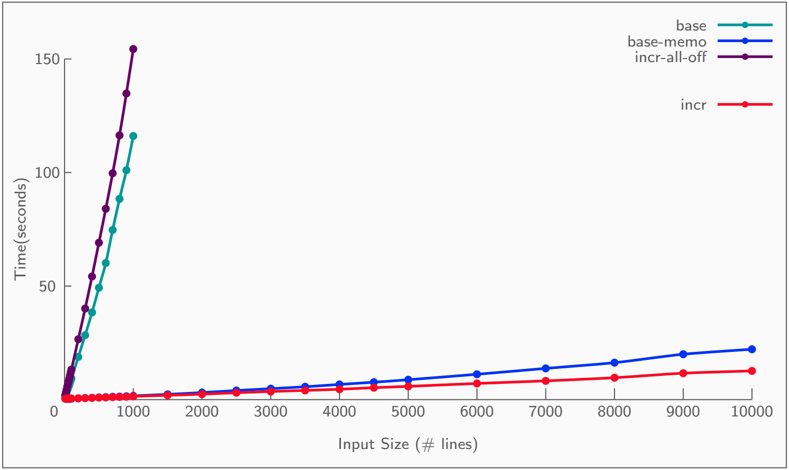
Experimental setup: structures populated with snippets of Joyce's *Ulysses*; 1.4GHz Intel Pentium III, 2GB RAM, SuSE Linux OS kernel 2.6.16; execution times collected from POSIX functions.

Experiments: Address Book Validator



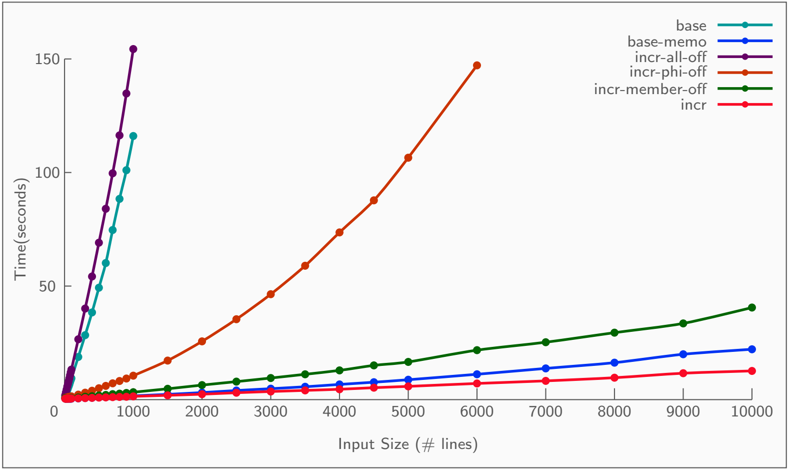
States	Formulas		Sat		Trees	
312	107517	99.8%	25727	99.9%	156615	42.1%

Experiments: Address Book Validator



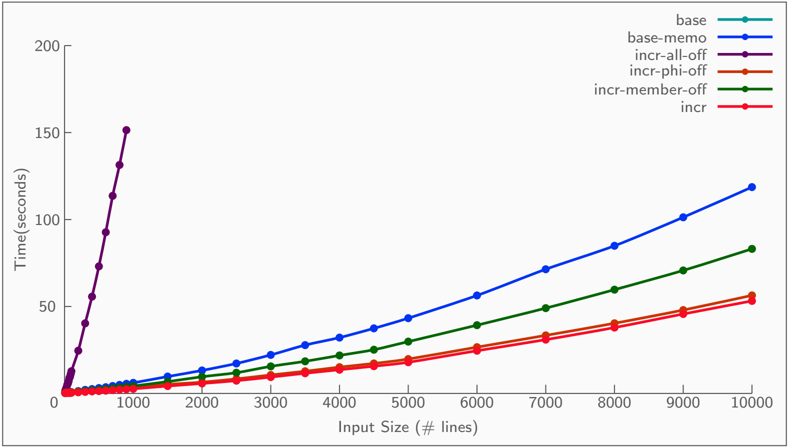
States	Formulas		Sat		Trees	
312	107517	99.8%	25727	99.9%	156615	42.1%

Experiments: Address Book Validator



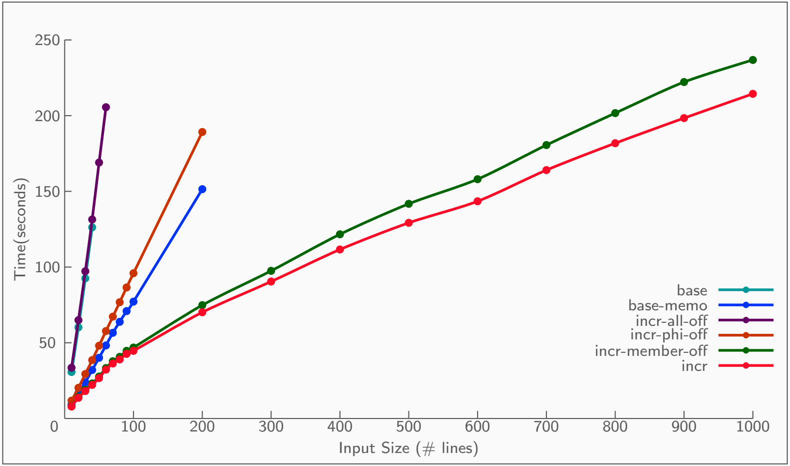
States	Formulas		Sat		Trees	
312	107517	99.8%	25727	99.9%	156615	42.1%

Experiments: Structured Text Parser



States	Formulas		Sat		Trees	
105	12461	99.1%	222	92.8%	3507706	81.4%

Experiments: iCalendar Lens



States	Formulas		Sat		Trees	
361	116939	97.4%	17600	87.8%	407652	76.5%

Related Work

Types and Automata:

- TQL [Cardelli and Ghelli, ESOP '01]
- "A Logic You Can Count On" [Dal Zilio, Lugiez, Meyssonnier, POPL '04]
- "Counting In Trees For Free" [Seidl, Schwentick, Muscholl, Habermehl, ICALP '04]
- Survey and Foundations: [Boneva and Talbot, RTA '05, LICS '05]

Implementations:

- "Static Checkers for Tree Structrures and Heaps" [Hague '04]
- "Boolean Operations and Inclusion Test for Attribute Element Constraints" [Hosoya and Murata, ICALP '03]

Conclusions and Future Work

Summary

- Strategies and algorithms;
- Implemented in Harmony;
- Reasonable performance.

Tune algorithm, hash-consing, memoization parameters.

Determinize sheaves formulas.

Implement Presburger arithmetic directly, optimized for adding constraints incrementally; also restricted fragments.

Extend to new structures and types: multitrees, ordered trees, also horizontal recursion, adjoint operators, etc.

Acknowledgements

Haruo Hosoya, Christian Kirkegaard, Stéphane Lescuyer, Thang Nguyen, Val Tannen, Penn PLClub and DB Group.

http://www.seas.upenn.edu/~harmony/