coreStar

Matko Botinčan, Dino Distefano, Mike Dodds, Radu Grigore,
Daiva Naudžiūnienė, Matthew J. Parkinson

August 2, 2011

Separation Logic—Models

$$\sigma, h \models \phi$$

where

$$\sigma \in \text{Variable} \rightharpoonup \text{Value} \tag{1}$$

$$h \in (\text{Value} \times \text{Variable}) \rightharpoonup \text{Value} \tag{2}$$

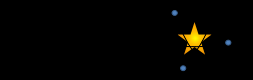
Examples

$$\sigma, [] \models \mathbf{emp} \tag{3}$$

$$[x : 0], h \models x = 0 \tag{4}$$

$$[x : 0], [(0, f) : 1] \models x \overset{f}{\mapsto} 1 \tag{5}$$

$$[x : 0, y : 1], [(0, f) : 2, (1, g) : 3] \models (x \overset{f}{\mapsto} 2 * y \overset{g}{\mapsto} 3) \wedge (x \neq y) \tag{6}$$



Separation Logic—Symbolic Execution of Calls

statement	$z := f(e)$	(7)
specification	$\{P\}\ x := f(y)\ \{Q\}$	(8)

Pure Only State Π

If $\Pi \Rightarrow P[e/y]$ is valid
then continue with $Q[z/x] \wedge \exists z'.\Pi[z'/z]$
else signal an error.

Spatial State Σ

If $\Sigma \Rightarrow (P[e/y] * F)$ is valid for *some* frame F
then continue with $Q[z'/z][z/x] * \exists z'.F[z'/z]$
else signal an error

Some Existing Tools that use Separation Logic

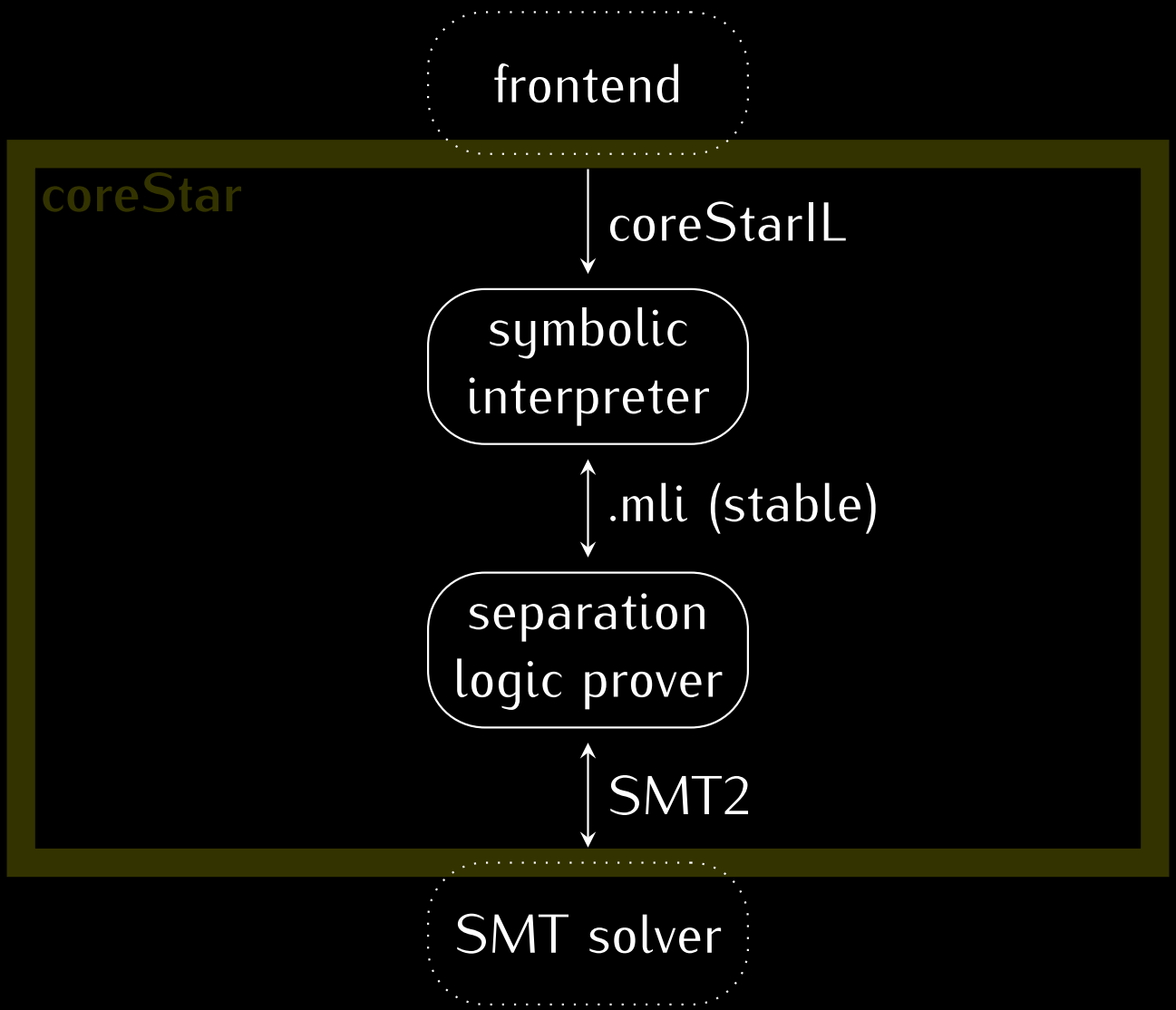
Not based on coreStar

- Smallfoot [Berdine et al., 2005]
- nameless [Chin et al., 2007]
- SpaceInvader [Yang et al., 2008],
  Abductor [Calcagno et el., 2009],
  Infer [Calcagno, Distefano, 2011]
- VeriFast [Jacobs et al., 2010]

Based on coreStar

- jStar [Distefano, Parkinson, 2008]
- MultiStar [van Staden, Calcagno, 2010]
- VMC [Botinčan et al. 2011]

coreStar Architecture

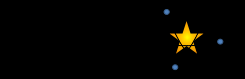


coreStarIL

$$Q ::= \{\phi\}\ S_1; \ldots S_n;\ \{\phi\} \tag{9}$$
$$S ::= x_1, \ldots, x_n := \{\phi\}\{\phi\} \tag{10}$$
$$S ::= \texttt{label}\ l \tag{11}$$
$$S ::= \texttt{goto}\ l_1, \ldots, l_n \tag{12}$$

$$\phi ::= \Pi \mid \Sigma \mid \phi * \phi \tag{13}$$
$$\Pi ::= \texttt{true} \mid E = E \mid E \neq E \mid p(E_1, \ldots, E_n) \tag{14}$$
$$\Sigma ::= \texttt{emp} \mid s(E_1, \ldots, E_n) \tag{15}$$

$$E ::= \ldots \tag{16}$$



coreStarIL—Example Frontend Translation

$$x = E \quad \rightsquigarrow \quad x := \{\}\{\mathrm{ret}_1 = E\} \tag{17}$$

$$x = *E \quad \rightsquigarrow \quad x := \{\mathtt{pto}(E, \_v)\}\{\mathtt{pto}(E, \_v) * \mathrm{ret}_1 = \_v\} \tag{18}$$

$$*E = F \quad \rightsquigarrow \quad () := \{\mathtt{pto}(E, \_v)\}\{\mathtt{pto}(E, F)\} \tag{19}$$

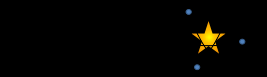
$$\mathtt{new}(x) \quad \rightsquigarrow \quad () := \{\}\{\mathtt{pto}(x, \_v)\} \tag{20}$$

$$\mathtt{del}(x) \quad \rightsquigarrow \quad () := \{\mathtt{pto}(x, \_v)\}\{\} \tag{21}$$

$$\mathtt{if}\,(E)\,\mathtt{then}\,B_1\,\mathtt{else}\,B_2$$

$$\rightsquigarrow$$

$$\begin{array}{l} \mathtt{goto}\, l_1, l_2; \\ \mathtt{label}\, l_1; () := \{\}\{E\}; B_1; \mathtt{goto}\, l_3; \\ \mathtt{label}\, l_2; () := \{\}\{E\}; B_2; \mathtt{label}\, l_3; \end{array} \tag{22}$$



Symbolic Interpreter

Queries

Given $\{P\}\ S_1; \ldots S_n;\ \{Q\}$,

- is it correct?
- is there *some* F such that $\{P * F\}\ S_1; \ldots S_n;\ \{Q\}$ is correct?

How It Works (for the First Query)

for each symbolic state (ϕ, s)
  ask the prover to generate candidate frames F
  for each frame F
    generate a next symbolic state
    (perhaps) apply user defined abstraction rules
    remember the new (abstract symbolic) state
repeat until fix-point

Separation Logic Prover

Queries

Given formulas ϕ and ψ,

- is $\phi \Rightarrow \psi$ valid?
- is there some F such that $\phi \Rightarrow (\psi * F)$ is valid?
- is there some A and some F such that $(\phi * A) \Rightarrow (\psi * F)$ is valid?

What It Does

- congruence closure (with uninterpreted functions)
- backtracking search based on user defined logic rules
- off-load pure goals to SMT solver

coreStarIL—Example Frontend Rules

```
rule pto_remove1:
| pto(?x, ?v)  |-  pto(?x, ?w)
without
  ?v != ?w
if
  pto(?x, ?v) |  |-  ?v = ?w


rule pto_pto_contradiction1 :
  pto(?x, ?v) * pto(?x, ?w) |  |-
if
```

Future

- issues we know of
  - interprocedural analysis
  - handle recursion (now off-loaded to front-end)
  - systematic way of proving soundness of logic rules and abstraction rules
  - abstraction rules not enough for some abstractions
  - baby APIs for symbolic and interpreter and for prover
  - documentation (*actually* re-implement Smallfoot, ...)
  - bug fixes
  - code cleanup
  - ...
- issues we don't know of
- please try it and improve it
  - `http://jstarverifier.org/`
  - `http://github.com/seplogic/corestar`

eof