Extending over-approximating models:

an introduction to mixed models and separation logic

David Schmidt

Kansas State University
(visiting École Polytechnique)

www.cis.ksu.edu/~schmidt

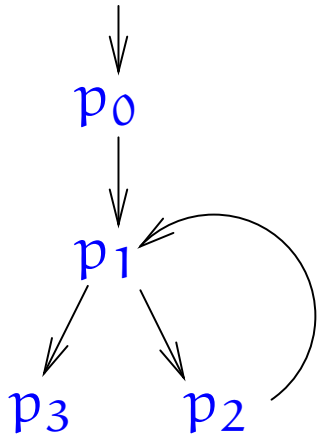
Outline

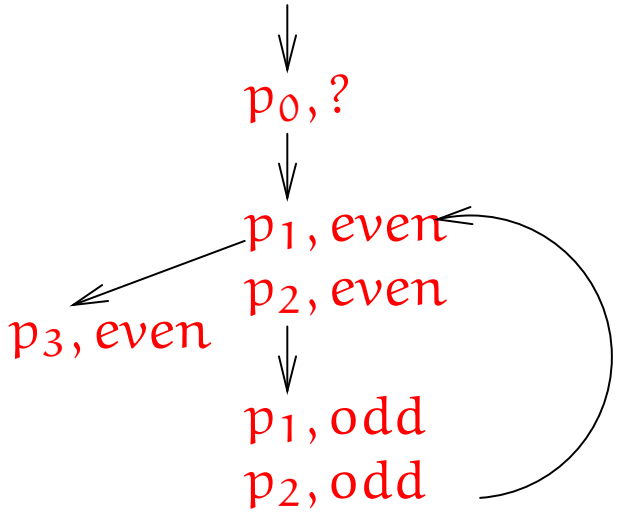
1. Review *over-approximating* relational models of control and data
2. Note that we can validate only *universal* properties on such models
3. Extend the models with *under-approximating relations* so that we can validate existentials. This gives us a *mixed model*.
4. Introduce modular validation, with an implicit existential, via *separation logic*
5. Note that the above approaches are *branching-time*, and that there is an equally interesting *linear-time* approach to validation

Models extracted from working hardware/software are *overapproximations*

Here is a program, its control-¤ow graph, and its polarity abstract interpretation:

p_0 : `x = 2;`
p_1 : `while x != 0 {`
p_2 : `x = x-1; }`
p_3 : *exit*





The program's concrete execution trace,

$$p_0, ? \to p_1, 2 \to p_2, 2 \to p_1, 1 \to p_2, 1 \to p_1, 0 \to p_3, 0$$

is modelled by abstract traces embedded in each of the two models.

The models use smaller state spaces but coarser transition relations.

A *labelled Kripke transition system* abstracts a process

$\langle \Sigma, \{\tau_\ell \subseteq \Sigma \times \Sigma \mid \ell \in \mathrm{Label}\}, \mathcal{I}_\Sigma : \Sigma \to \mathcal{P}(\mathrm{Atom}) \rangle$

$\Sigma = \{s0, s1, s2\}$

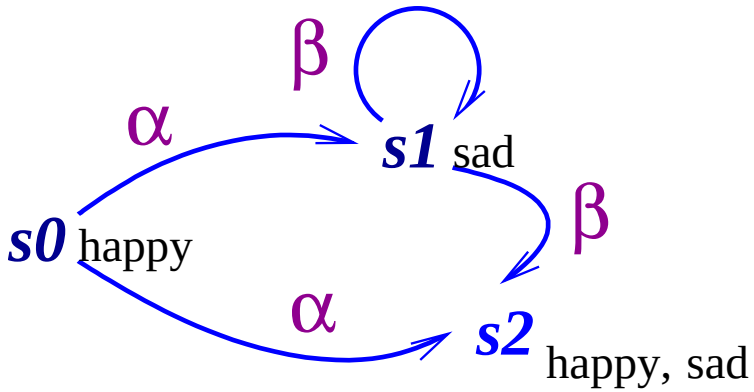
$\tau_\alpha = \{(s0, s1), (s0, s2)\}$

$\tau_\beta = \{(s1, s1), (s1, s2)\}$

$\mathcal{I}_\Sigma(s0) = \{\mathrm{happy}\}$

$\mathcal{I}_\Sigma(s1) = \{\mathrm{sad}\}$

$\mathcal{I}_\Sigma(s2) = \{\mathrm{happy}, \mathrm{sad}\}$



There are two transition relations, τ_α and τ_β.

A Kripke system might be approximated further:
$A = \{\bot, a0, a12, \top\}$

$\alpha\{\} = \bot$

$\alpha\{c0\} = a0$

$\alpha\{c1\} = a12 = \alpha\{c2\} = \alpha\{c1, c2\}$

$\alpha S = \top$, otherwise

$\gamma(\bot) = \{\}$

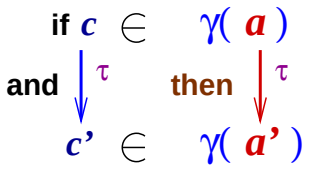
$\gamma(a0) = \{c0\}$

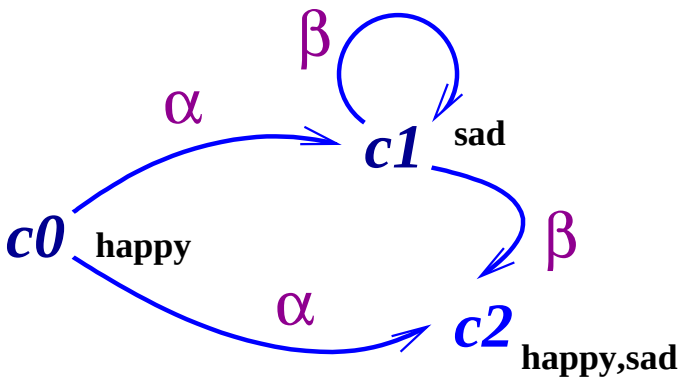
$\gamma(a12) = \{c1, c2\}$

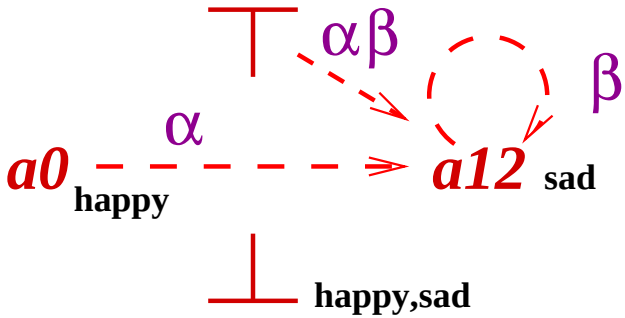
$\gamma(\top) = \{c0, c1, c2\}$

$\mathcal{I}_A(a) = \cap\{\mathcal{I}_C(c) \mid c \in \gamma(a)\}$

α, γ form a *Galois connection*, induce the simulation relation,
$\rho \subseteq \Sigma \times A$ — $c\ \rho\ a$ iff $c \in \gamma(a)$ — and de£ne coarser α and β.







Graph models apply to storage shapes

$$\Sigma = \{c0, c1, c2\}$$

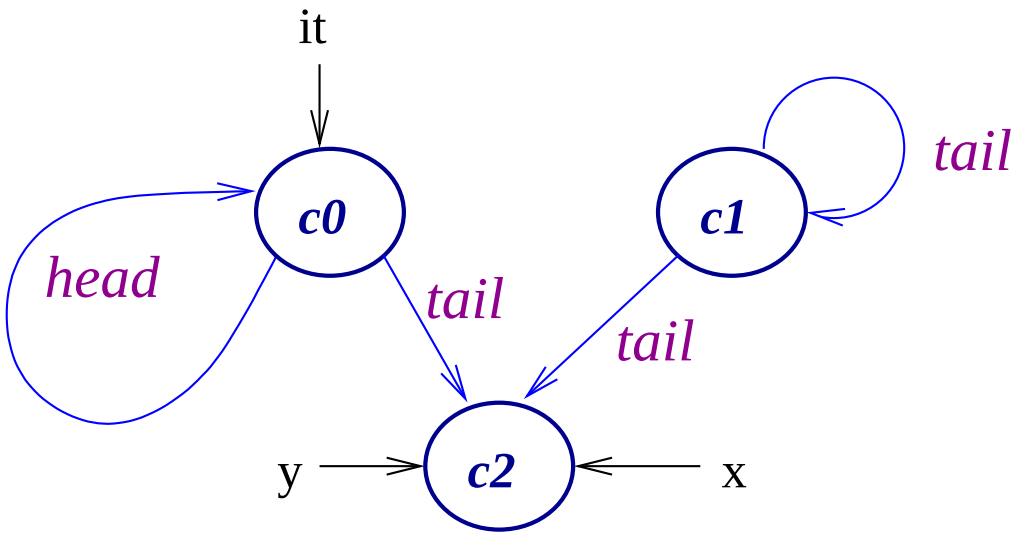
$$\tau_{head} = \{(c0, c0)\}$$

$$\tau_{tail} = \{(c0, c2), (c1, c1), (c1, c2)\}$$

$$\mathcal{I}_\Sigma(c0) = \{it\}$$

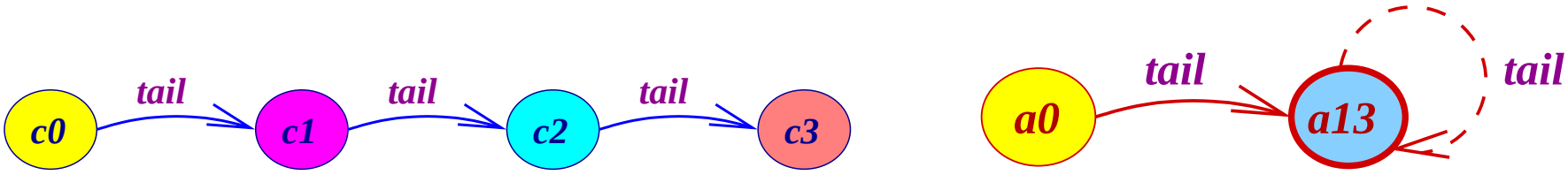
$$\mathcal{I}_\Sigma(c1) = \{\}$$

$$\mathcal{I}_\Sigma(c2) = \{x, y\}$$



Rather than states, the nodes now represent *cells*.

A linear list can be abstracted to its “shape graph”:



The noncircular list is modelled by a more compact representation that distinguishes only between the head and non-head nodes.

A data base can be abstracted by forgetting a dimension:

surname	*name*	*eye color*
Ricardo	Ricky	brown
Ricardo	Lucy	blue
Mertz	Fred	green

surname	*eye color*
Ricardo	brown ⊔ blue
Mertz	green

The abstraction produces a smaller element set with loss of precision.

Speci£cations/invariants are *underapproximations* of correct behaviours

An acceptable database must associate a unique eye color to each person:

$$\forall p : \mathrm{Person}, \exists! c : \mathrm{Color}, c \in p.\mathrm{eyeColor}$$

An acceptable list must be noncircular:

$$\forall n \geq 0, (\forall i \in 0..n-1, a_{i+1} = a_i.\mathrm{tail}) \wedge (\forall i, j \in 0..n, i \neq j \Rightarrow a_i \neq a_j)$$

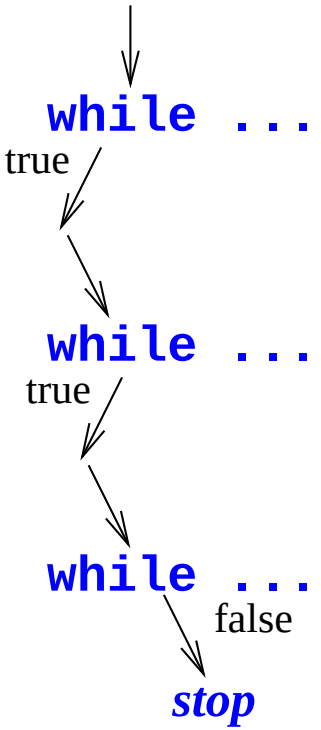
An acceptable program execution must terminate:

$$p_0 \models^{\forall} \mathrm{F}(\mathrm{at}\ p_3)$$

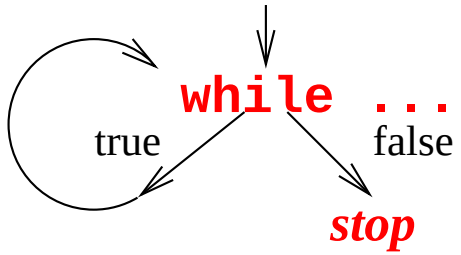
Alas, *none* of these properties can be proved for the overapproximating models we just saw!

Overapproximating models add extra transitions that generate nonexecuted paths. These paths reduce the properties we can prove on the overapproximating model (that also hold true for the concrete structure being modelled).

Concrete trace:



Overapproximation:



On the overapproximation, we cannot prove

∀ paths,

termination occurs

although it holds for the concrete semantics,

and we cannot *safely* prove

∃ path that is in£nite

because it does *not* hold for the concrete semantics

So, what can we verify?

Given the α, γ machinery, we can soundly validate universal properties of nodes, a, in overapproximating models, M:

$$\phi \quad ::= \quad p \quad | \quad \phi \wedge \phi \quad | \quad \phi \vee \phi \quad | \quad \forall R.\phi \quad | \quad \mu Z.\phi \quad | \quad \nu Z.\phi \quad | \quad Z$$

(Note: This is a fragment of *description logic*.)

When $a \models \phi$, then for all $c \in \gamma(a)$, we have $c \models \phi$.

$a \models \phi$ iff $p \in \mathcal{I}_\Sigma(a)$

$a \models \phi_1 \wedge \phi_2$ iff $a \models \phi_1$ and $a \models \phi_2$

$a \models \forall R.\phi$ iff for all $a' \in a.R^\sharp,\ a' \models \phi$ (where $R^\sharp$ is sound w.r.t. α, γ)

$a \models \mu Z.\phi$ iff there exists $i \geq 0$ such that $a \models \phi_i$, where $\begin{cases} \phi_0 = false \\ \phi_{i+1} = [\phi_i/Z]\phi \end{cases}$

$a \models \nu Z.\phi$ iff for all $i \geq 0, a \models \phi_i$, where $\begin{cases} \phi_0 = true \\ \phi_{i+1} = [\phi_i/Z]\phi \end{cases}$

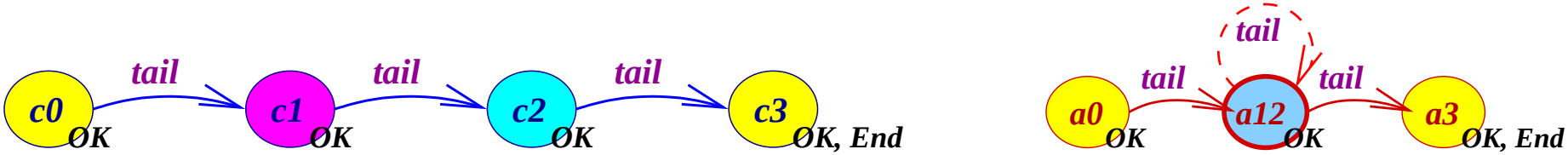
Note: The de£nitions of μ and ν assume that M is £nite.

$a \models \phi_1 \wedge \phi_2$ iff $a \models \phi_1$ and $a \models \phi_2$

$a \models \forall R.\phi$ iff for all $a' \in a.R^\sharp,\ a' \models \phi$ (where $R^\sharp$ is sound w.r.t. α, γ)

$a \models \mu Z.\phi$ iff there exists $i \geq 0$ such that $a \models \phi_i$, where $\begin{cases} \phi_0 = false \\ \phi_{i+1} = [\phi_i/Z]\phi \end{cases}$

$a \models \nu Z.\phi$ iff for all $i \geq 0, a \models \phi_i$, where $\begin{cases} \phi_0 = true \\ \phi_{i+1} = [\phi_i/Z]\phi \end{cases}$



Let $tail \subseteq Node \times Node$ be the transition relation.

We can safely prove:

$a_0 \models \nu Z.isOK \wedge \forall tail.Z$: all nodes reached from a_0 (and so, c_0) are OK

$a_3 \models \mu Z.atEnd \vee \forall tail.Z$: all paths from a_3 (and so, c_3) terminate

We cannot prove: $a_0 \models \mu Z.atEnd \vee \forall tail.Z$ (but it holds for c_0)

We cannot safely prove:

$a_{12} \models \nu Z.\exists next.Z$: there is a cycle (because this fails for c_1 and c_2)

Why does this matter?

Program and data-structure models are almost always overapproximations. The speci£cations/invariants that we wish to validate often require a richer logic than the one just presented.

Can we do better?

1. Make the overapproximating model more detailed by adding *underapproximating relations*, dual to the ones we have. We check both universal and existential properties on the *mixed model*, and there are some specialization tricks (*generalized model checking, "focus" operations*, etc.) to improve precision.

2. Retain just the overapproximating model but add some key operators on relations that enrich the language of properties we might prove. This is a basis of *separation logic*.

1. Mixed models

An overapproximating model uses "maybe" relations, $R^\sharp$:

$(a, b) \in R^\sharp$, that is,

$$a \overset{R}{\dashrightarrow} b$$

means "a may be related to b" in a concretization of this model.

(Precisely, $(u, v) \in R$ and $u \in \gamma(a)$ imply $(a, b) \in R^\sharp$ and $v \in \gamma(b)$.) When $(a, b) \notin R^\sharp$, it signi£es that *no* concretization of the model relates a concretization of a to b.

$$a \models \forall R.\phi \text{ iff for all } b \in a.R^\sharp,\ b \models \phi$$

The dual is a "must" relation: $(a, b) \in R^\flat$:

$$a \overset{R}{\longrightarrow} b$$

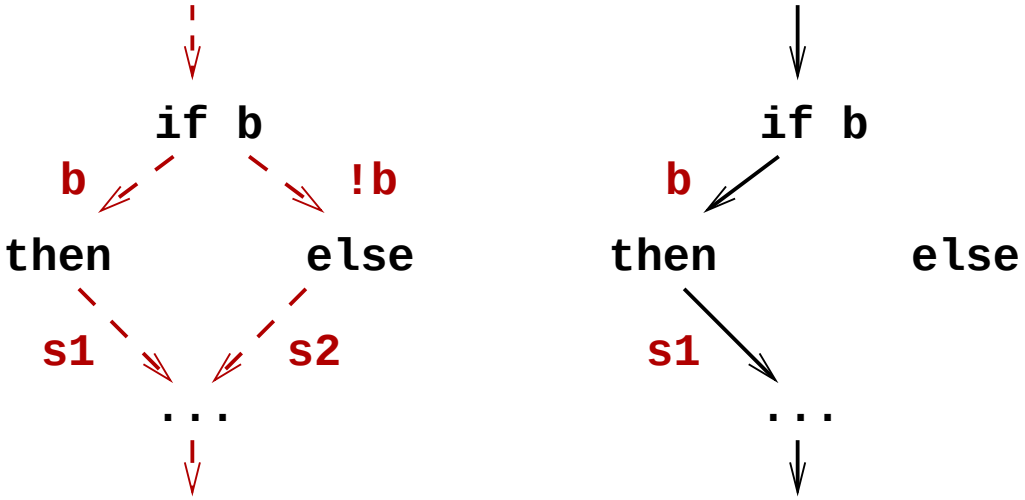
means "a must be related to b" in every concretization of this model.

(Precisely, $(a, b) \in R^\flat$ and $u \in \gamma(a)$ imply $(u, v) \in R$ and $v \in \gamma(b)$.)

$$a \models \exists R.\phi \text{ iff there exists } b \in a.R^\flat,\ b \models \phi$$

What does a “maybe” transition denote?

what may *possibly* execute:

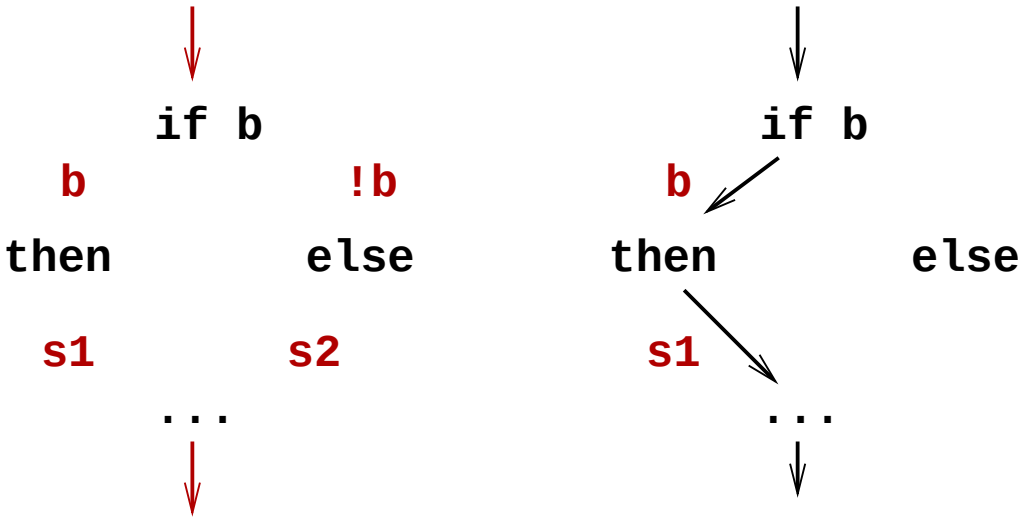


what may *possibly* be pointed:

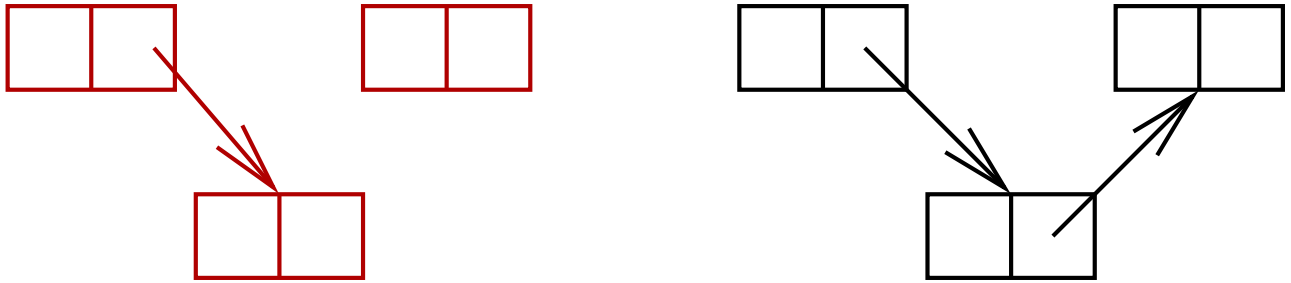
$a \models \forall R.\phi$ iff for all $a' \in a.R^\sharp$, $a' \models \phi$

What does a “must” transition denote?

what must *necessarily* execute:



what must *necessarily* be linked:

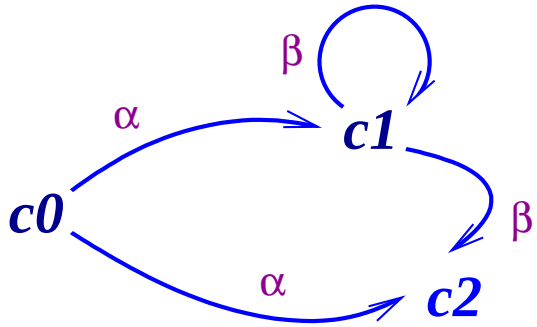


$a \models \exists R.\phi$ iff there exists $a' \in a.R^\flat,\ a' \models \phi$

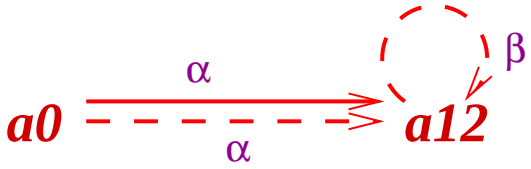
We can validate a full description logic on a MTS

We validate universals on the overapproximated relations and existentials on the underapproximated ones. And we can validate negations via *refutations* on the *dual* approximation.

Example:



abstracts to



$$a0 \models^{under} \exists\alpha.\forall\beta.\neg at\_zero$$

iff $a12 \models^{over} \forall\beta.\neg at\_zero$

iff $a12 \models^{over} \neg at\_zero$

iff $a12 \not\models^{under} at\_zero$

iff $true$

This is fun, but extracting mixed models is hard work! (The reason is that state-abstraction is almost always overapproximating, making it dif£cult to extract *under*approximating relations.)

Modular validation

Given a model, $\langle \Sigma, \{\tau_\ell\}_{\ell \in L} \rangle$, and $a \in \Sigma$, a judgement $a \models \phi$ is checked with respect to the entire model, that is,

$$a, \langle \Sigma, \{\tau_\ell\}_{\ell \in L} \rangle \models \phi$$

Is there a more "modular" variant of property checking, where a subset of the model is used to validate ϕ?

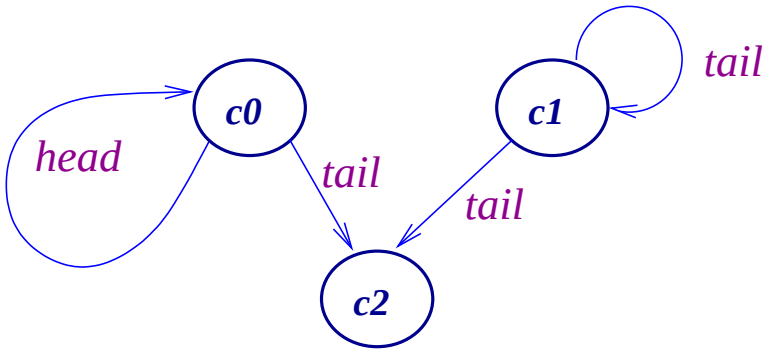
That is, we want to divide the model into disjoint "regions," h_i, so that we can use this reasoning principle:

$$\frac{h_1 \models \phi_1 \quad h_2 \models \phi_2 \quad h_1 \# h_2}{h_1 \circ h_2 \models \phi_1 * \phi_2}$$

where $h_1 \# h_2$ asserts that h_1 and h_2 are disjoint regions. $\phi_1 * \phi_2$ expresses an assertion whose conjuncts hold true for disjoint regions — *no aliasing/sharing of region domains allowed!*

This is the motivation for *separation logic*.

The standard application is to storage heaps, e.g.,
$h = \langle Cell, \{head, tail\}\rangle$. (Say that $domain(h) = Cell$.)



We say that $\langle Cell_1, \{head_1, tail_1\}\rangle \# \langle Cell_2, \{head_2, tail_2\}\rangle$ iff $Cell_1 \cap Cell_2 = \{\}$.

The composition of two regions is

$\langle Cell_1, \{head_1, tail_1\}\rangle \circ \langle Cell_2, \{head_2, tail_2\}\rangle =$

$\langle Cell_1 \cup Cell_2, \{head_1 \cup head_2, tail_1 \cup tail_2\}\rangle$

For example, the above model can be divided as follows:

$$\langle \{c_0\}, \{\{(c_0, c_0)\}, \{(c_0, c_2)\}\}\rangle \# \langle \{c_1, c_2\}, \{\{\}, \{(c_1, c_2), (c_2, c_2)\}\}\rangle$$

The technique can also be used to organize a set of parallel processes.

2. Separation logic (O'Hearn, Pym, Yang)

Additives (the usual classical logic): Let $h \in Heap$:

$h \models true$ always

$h \models p \wedge p'$ iff $h \models p$ and $h \models p'$

$h \models \exists x.p_x$ iff exists $v \in \Sigma$ s.t. $h \models p_v$

$h \models E_1 = E_2$ iff $[\![E_1]\!] = [\![E_2]\!]$

Multiplicatives (based on a commutative partial monoid, $(Heap, \circ, \epsilon)$):

$h \models emp$ iff $h = \epsilon$

$h \models p * p'$ iff there exist h_0, h_1 such that $h_0 \# h_1$,
$h = h_0 \circ h_1$, $h_0 \models p$, and $h_1 \models p'$

$h \models p \mathrel{-\!\!*} p'$ iff for all h', if $h' \# h$ and $h' \models p$,
then $h \circ h' \models p'$

$h \models R(E_1, E_2)$ ... application dependent ...

Separation logic proves correctness properties

Because we can assert disjointness, we can write an assertion that de£nes a (tail-)noncircular list:

$$\mathtt{nc}(\ell) \text{ iff}_{lfp} \ (\ell = \mathtt{null}) \ \vee \ (\exists \mathtt{c}.\ tl(\ell, \mathtt{c}) * \mathtt{nc}(\mathtt{c}))$$

where $\mathtt{h} \models tl(\mathtt{a}, \mathtt{b})$ iff $\mathtt{domain}(\mathtt{h}) = \{\mathtt{a}\}$ and $(\mathtt{a}, \mathtt{b}) \in tail \in \mathtt{h}$.

The star ($*$) ensures that *all cells in the list's tail are disjoint from the list's head, addressed by* ℓ. We might prove that a copy function constructs a noncircular list:

```
copy(Cell x )  {
  nc(x)
    y := (if x = null
            then x
            else new Cell(x.hd, copy(x.tl)));
  nc(y)
    return y; }
```

The Reynolds-O'Hearn "small" axioms

$\mathtt{E} \doteq \mathtt{E}'$ abbreviates $\mathtt{E} = \mathtt{E}' \wedge emp$ (where emp is the "empty" heap).

$\mathtt{E}_1 \mapsto \mathtt{E}_2, \mathtt{E}_3$ abbreviates $hd(\mathtt{E}_1, \mathtt{E}_2) \wedge tl(\mathtt{E}_1, \mathtt{E}_3)$.

Assume that $\mathtt{x}$, $\mathtt{a}$, $\mathtt{b}$, and $\mathtt{c}$ are *variables* and that $\mathtt{x} \notin \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$

The small axioms for command forms are

$$\{\mathtt{x} \doteq \mathtt{a}\}\ \ \mathtt{x} := \mathtt{E}\ \ \{\mathtt{x} \doteq \mathtt{E}[\mathtt{a}/\mathtt{x}]\}$$

$$\{\mathtt{E} \mapsto \mathtt{a}, \mathtt{b}\}\ \ \mathtt{E.tl} := \mathtt{E}'\ \ \{\mathtt{E} \mapsto \mathtt{a}, \mathtt{E}')\}$$

$$\{\mathtt{x} \doteq \mathtt{a}\}\ \ \mathtt{x} := \mathtt{new\ C}(\mathtt{E}_1, \mathtt{E}_2)\ \ \{\mathtt{x} \mapsto \mathtt{E}_1[\mathtt{a}/\mathtt{x}], \mathtt{E}_2[\mathtt{a}/\mathtt{x}]\}$$

$$\{\mathtt{x} = \mathtt{a} \wedge \mathtt{E} \mapsto \mathtt{b}, \mathtt{c}\}\ \ \mathtt{x} := \mathtt{E.tl}\ \ \{\mathtt{x} = \mathtt{c} \wedge \mathtt{E}[\mathtt{a}/\mathtt{x}] \mapsto \mathtt{b}, \mathtt{c}\}$$

A Hoare triple, $\{\phi\}\mathtt{C}\{\psi\}$, de£nes a relation over commands, *but it also de£nes "command-transfer functions" that might be used by a data-¤ow algorithm (e.g., Whaley-Rinard's analysis, to come).*

Structural rules for inference

The small axioms gain utility when used with these structural rules; let $modified(S)$ be those variables that are targets of assignments in S.

$$\text{Consequence}: \quad \frac{p \supset p' \;\; \{p'\}S\{q'\} \;\; q' \supset q}{\{p\}S\{q\}}$$

$$\text{Subst}: \quad \frac{\{p\}S\{q\}}{(\{p\}S\{q\})[E_1/x_1, \cdots E_k/x_k]} \quad \text{where } \{x_1, \cdots, x_k\} \supseteq free(p, S, q),$$

and $x_i \in modified(S)$ implies E_i is a var, $E_i \notin free(E_j), j \neq i$

$$\text{Frame}: \quad \frac{\{p\}S\{p'\}}{\{p * q\}S\{p' * q\}} \quad \text{where } modified(S) \cap free(q) = \{\}$$

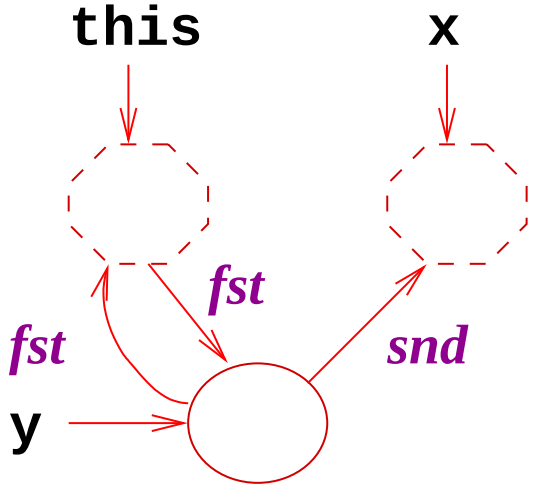
The Subst rule motivates the usual rule for procedure invocation (as substitution of actuals for formals).

The Frame rule embeds a result proved of a heap "region" into a larger heap, *justifying modular reasoning by reasoning on disjoint heap regions*.

Application of separation logic to Whaley-Rinard analysis

Consider this Whaley-Rinard shape analysis of `m(C x){...}`:

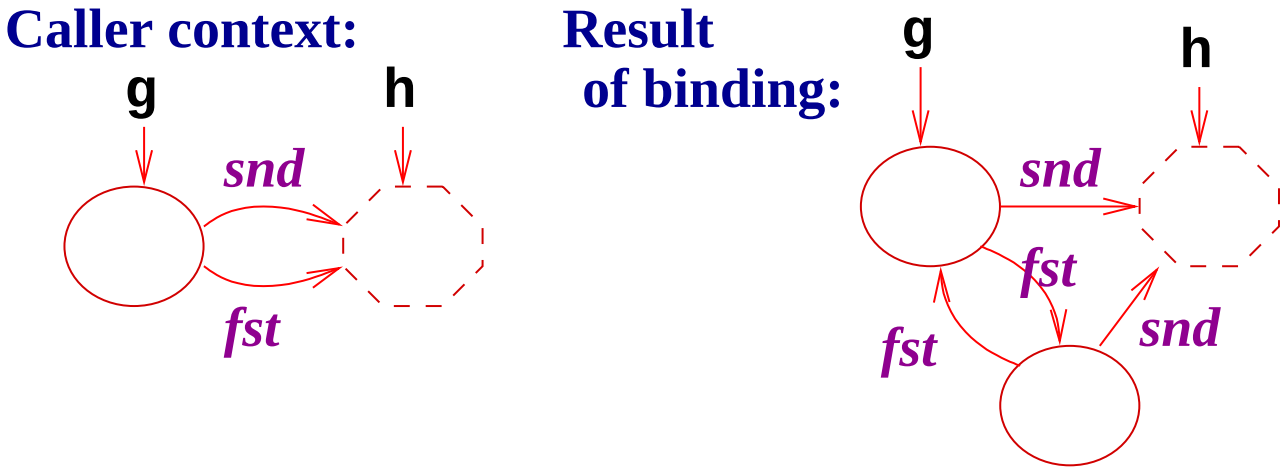
```
class C {
   C fst; C snd;
   function m(C x)  {
      C y := new C(this,x);
      this.fst := y; }
}
```



The shape analysis is neatly summarized as this Hoare triple:

$$\begin{array}{l} \texttt{m(C x)\{} \quad \{\texttt{this} \mapsto a, b\} \\ \qquad\qquad \texttt{C y} := \texttt{new C(this, x);}\ \ \texttt{this.fst} := \texttt{y;} \\ \qquad\qquad \{\exists \texttt{y}.\ (\texttt{this} \mapsto \texttt{y}, b) * (\texttt{y} \mapsto \texttt{this}, \texttt{x})\} \\ \qquad \texttt{\}} \end{array}$$

Say there is an invocation, `g.m(h)`:



The corresponding step in separation logic is

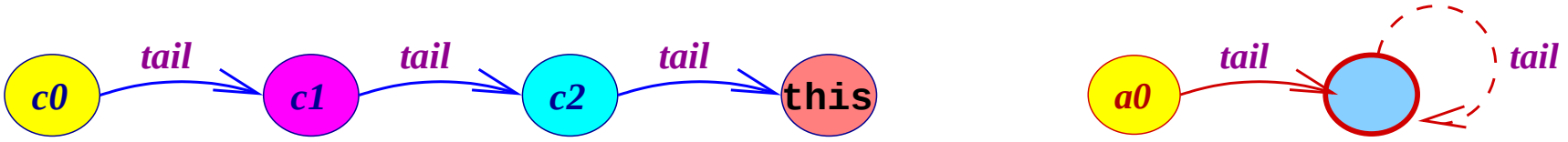
$$\{\mathtt{g} \mapsto \mathtt{h}, \mathtt{h}\} \quad \mathtt{g.m(h)} \quad \{\exists \mathtt{y}.(\mathtt{g} \mapsto \mathtt{y}, \mathtt{h}) * (\mathtt{y} \mapsto \mathtt{g}, \mathtt{h})\}$$

because $[\mathtt{this}/\mathtt{g},\ \mathtt{h}/\mathtt{x}]$ substitutes into the previous triple for `m(C x)`:

$$\{\mathtt{this} \mapsto a, b\}$$

$$\mathtt{C\ y} := \mathtt{new\ C(this, x);}\ \ \mathtt{this.fst} := \mathtt{y;}$$

$$\{\exists \mathtt{y}.\ (\mathtt{this} \mapsto \mathtt{y}, b) * (\mathtt{y} \mapsto \mathtt{this}, \mathtt{x})\}$$



There are many ad-hoc conventions for asserting noncircularity in shape graphs; separation logic gives us the convention for free. Consider

```
x := this;
while (...) {
  x:= new C(..., x); // \alpha names the summary object
}
```

The loop's approximate invariant, mechanically computed, reads

$$(\mathtt{x} = \alpha) \ \wedge \ (\alpha \mapsto ..., (\alpha^* \vee \mathtt{this}))$$

The expression abbreviates a sequence of *distinct objects*:

$$\exists i \geq 0.\, (\mathtt{x} = \alpha_i) \ \wedge \ ((\alpha_i \mapsto ..., \alpha_{i-1}) * (\alpha_{i-1} \mapsto ..., \alpha_{i-2}) * \cdots * (\alpha_0 \mapsto ..., \mathtt{this}))$$

The Sagiv-Reps-Wilhelm TVLA shape-analysis system uses exactly this interpretation of its summary nodes.

The existential nature of $*$

Recall that $\exists$-properties cannot be validated on overapproximating models. Nonetheless, separation logic's $*$ *embeds* a useful existential property into its assertions:

Read $h \models \phi_1 * \phi_2$ as *there exists* h_1, h_2 such that $h_1 \# h_2$, $h = h_1 \circ h_2$, $h_1 \models \phi_1$, and $h_2 \models \phi_2$.

The motivating example is

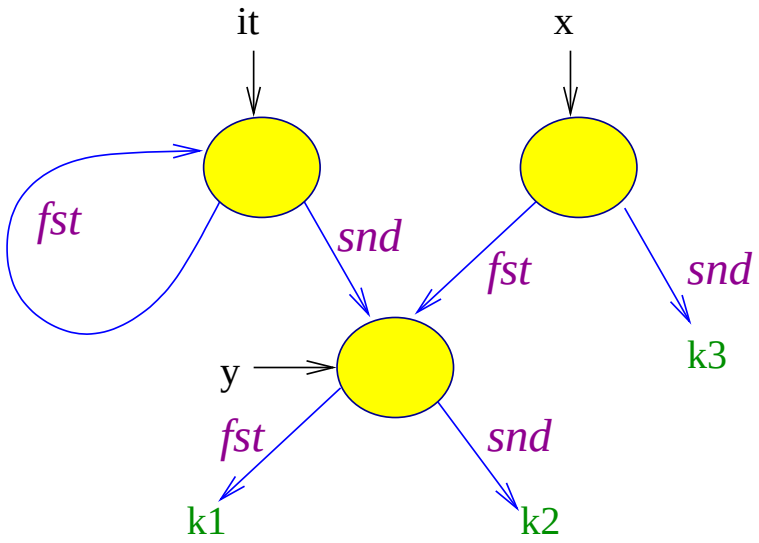
$$isAliased(c) \text{ iff } tl(\alpha, c) * tl(\beta, c)$$

where α and β are placeholders (Prolog-like logical variables) *that must denote distinct cells*.

$\#$ can be checked on abstract values: $\{a\}\#\{a'\}$ iff $a \sqcap a' \neq \bot$, assuming γ is strict. (Or, we can *assume*, as is done in TVLA, that $a \neq a'$ implies $\gamma(a) \cup \gamma(a') = \{\}$.)

“Store-less” models: Path sets

Jonkers and Deutsch proposed “storeless” (heap-less) models:



The heap shape is modelled by right-regular equivalence sets of paths from the “entry point,” $\mathtt{it}$:

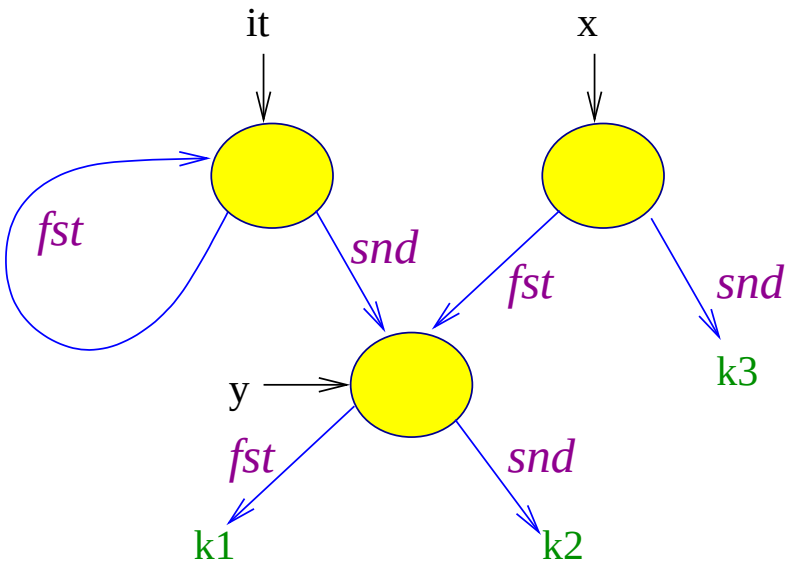
$$\{\mathtt{fst}^i \mid i \geq 0\} \qquad \{\mathtt{fst}^i.\mathtt{snd} \mid i \geq 0\}$$

$$\{\mathtt{fst}^i.\mathtt{snd}.\mathtt{fst} \mid i \geq 0\} \qquad \{\mathtt{fst}^i.\mathtt{snd}^2 \mid i \geq 0\}$$

Deutsch developed clever fsa over-approximations of the equivalence classes.

Blanchet's path models

Many questions regarding escapes, leaks, and aliases are answered by the paths from one object of interest to another, e.g., from a global variable to the heap's entry point:



$$\{y.\mathtt{snd}^{-1}.\mathtt{fst}^{i}.(\mathtt{fst}^{-1})^{j} \mid i,j \geq 0\}$$
$$\cup \{x.\mathtt{fst}.\mathtt{snd}^{-1}.\mathtt{fst}^{i}.(\mathtt{fst}^{-1})^{j} \mid i,j \geq 0\}$$

The paths have been normalized by the cancellation law,

$$\mathtt{fst}^{-1}.\mathtt{fst} \equiv \epsilon$$

The cancellation law gives the paths a pleasant, regular format.

The paths are *traces* through the heap, and questions about the traces can be asked in the language of *linear temporal logic*. Let π be a trace from variable $\mathtt{x}$ to $\mathtt{it}$, the result/heap-entry.

We can ask standard questions:

- Is part of $\mathtt{x}$ embedded in the result? $\pi \models \mathtt{at\_x} \wedge \mathrm{F}(\mathtt{des}^{-1})$
- Does $\mathtt{x}$'s cell itself escape in the result? $\pi \models \mathtt{at\_x} \wedge \mathrm{G}(\mathtt{des}^{-1})$
- Is part of $\mathtt{x}$ aliased to $\mathtt{y}$? $\pi \models \mathrm{F}(\mathtt{at\_y})$
- Is $\mathtt{x}$ a cyclic structure? $\pi \models \mathrm{GF}(\mathtt{at\_x})$

This approach has been extended by Bazga, Iosif, and Lakhnech.

My recent work

- Formalizing transition relations as functions of form, $A \to \mathcal{P}(A)$, where $\mathcal{P}$ is a lower- or upper-powerdomain, and characterizing simulations on such relations in terms of Galois connections on the powerdomains
- Applying the above to understand the logics that can be validated and refuted on over- and under-approximating models of relations
- Attempting to integrate separation logic's $*$ operator into modular abstract interpretations of overapproximating models, by studying Blanchet's modular escape analysis for ML.
- Also, Élodie Sims, a PhD student joint between Kansas and École Polytechnique, has formalized and proved correct an alias ("partitioning") analysis based on separation logic.

References

This talk: `www.cis.ksu.edu/~schmidt/papers`

1. B. Blanchet. Ph.D. thesis. `http://www.di.ens.fr/~blanchet`
2. M. Bozga, R. Iosif and Y. Lakhnech. Storeless Semantics and Alias Logic. Proc. PEPM 2003.
3. D. Dams, R. Gerth, O. Grumberg. Abstract Interpretation of Reactive Systems. *ACM TOPLAS* 19 (1997).
4. M. Huth, R. Jagadeesan, D. Schmidt. Modal transition systems: a foundation for three-valued program analysis. Proc. ESOP 2002.
5. C. Loiseaux, et al. Property preserving abstractions for the veri£cation of concurrent systems. *Formal Methods in System Design* 6 (1995).
6. P. O'Hearn, J. Reynolds, H. Yang. Local reasoning about programs that alter data structures. `http://www.dcs.qmul.ac.uk/~ohearn/`
7. D. Pym, P. O'Hearn, H. Yang. Possible worlds and resources: The semantics of BI. *Theoretical Computer Science* `http://www.dcs.qmul.ac.uk/~ohearn/`
8. M. Sagiv, T. Reps, R. Wilhelm. Parametric Shape Analysis via 3-Valued Logic. 26th ACM POPL, 1999.