Abstract Models of Shape
Branching- and Linear-Time

David Schmidt
Kansas State University

Outline

1. “branching time” models: graphs
2. abstraction of nodes via Galois connections; abstraction of arcs via simulations
3. under- and over-approximations; mixed- and modal-transition systems
4. assertion-based abstractions
5. “linear time” models: path sets and their abstraction

Labelled Kripke transition systems

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

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

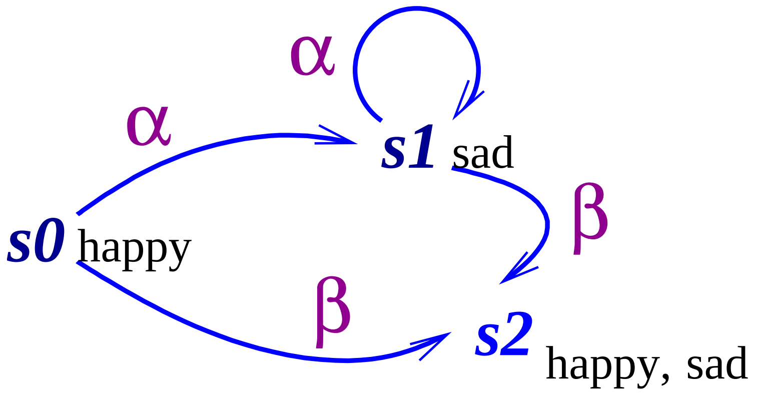
$$\tau_\alpha = \{(s0, s1), (s1, s1)\}$$

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

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

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

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



We might identify initial states, $\Sigma_0 \subseteq \Sigma$, also.

Graph models apply to storage shapes

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

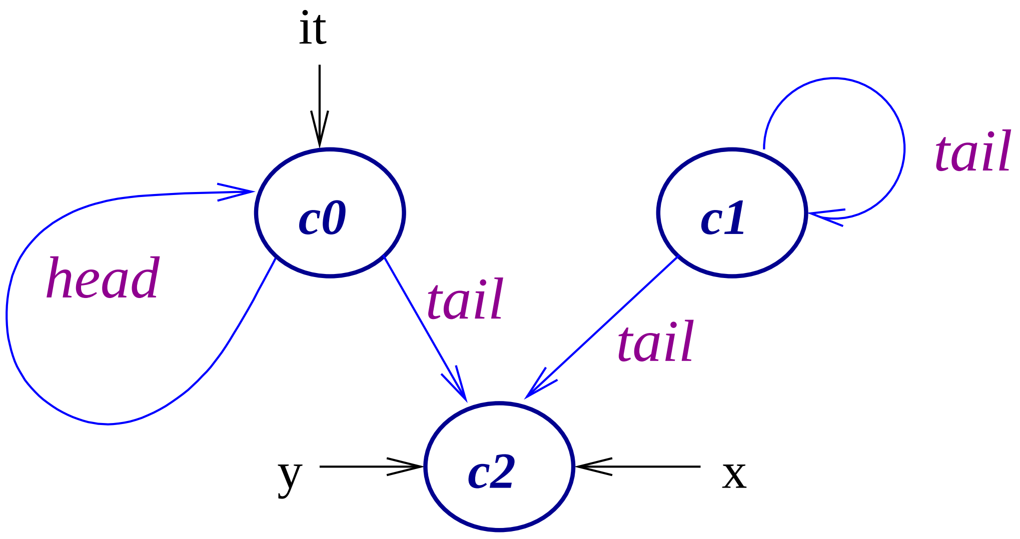
$$\tau_{head} = \{(c0, c0)\}$$

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

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

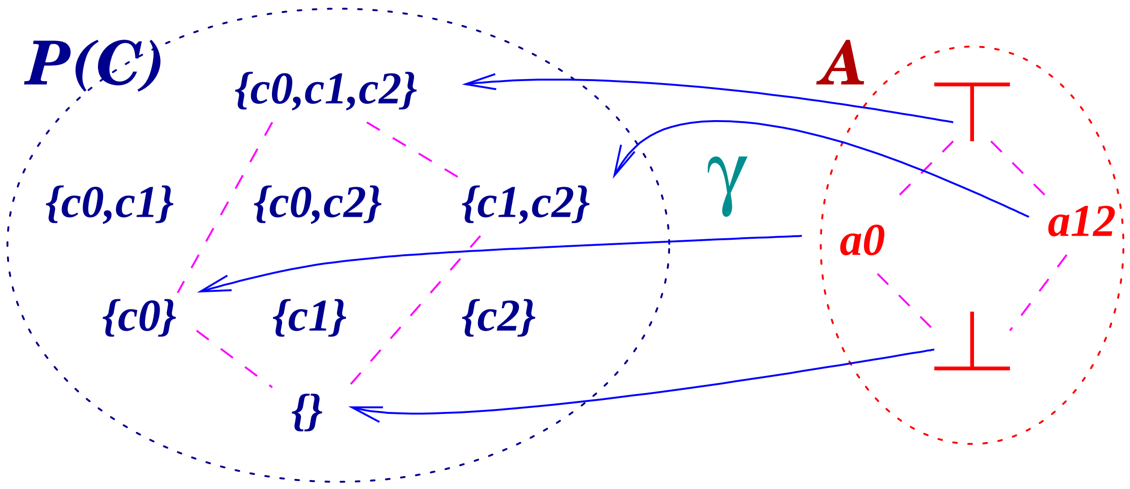
$$\mathcal{I}_\Sigma(c1) = \{\}$$

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



Rather than states, the nodes now represent cells.

A Galois Connection abstracts the nodes



Typically, A's elements are *(i)* correctness properties of interest, or *(ii)* subsets of Atom. Then, $\gamma : A \to \mathcal{P}(C)$ is de£ned so that $\gamma(a)$ produces all nodes/cells having property a.

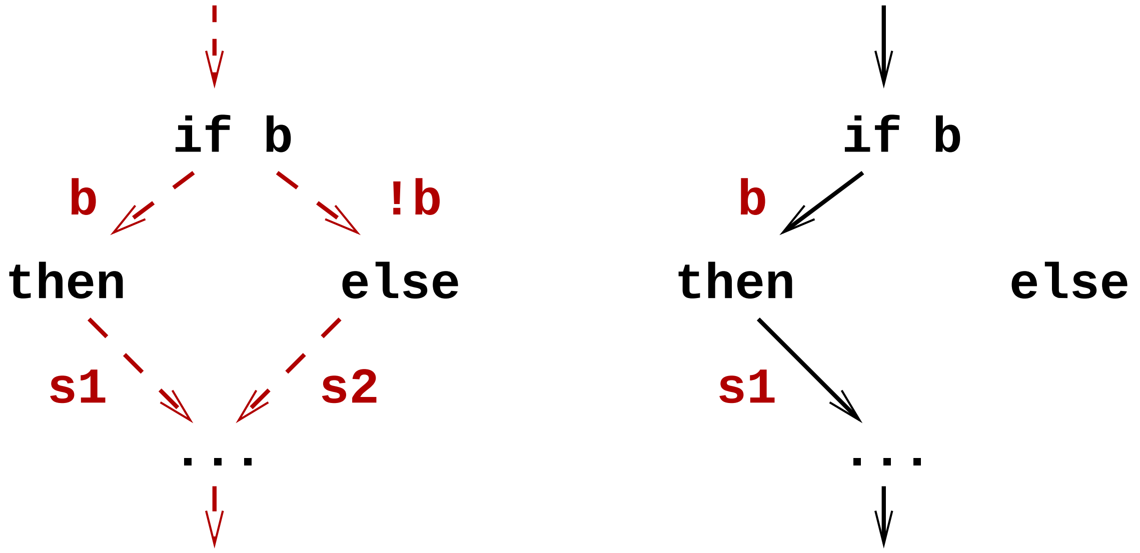
γ's image must be a Moore family:

- $\gamma(\top) = C$
- for all a, a', there exists a'' such that $\gamma(a) \cap \gamma(a') = \gamma(a'')$.

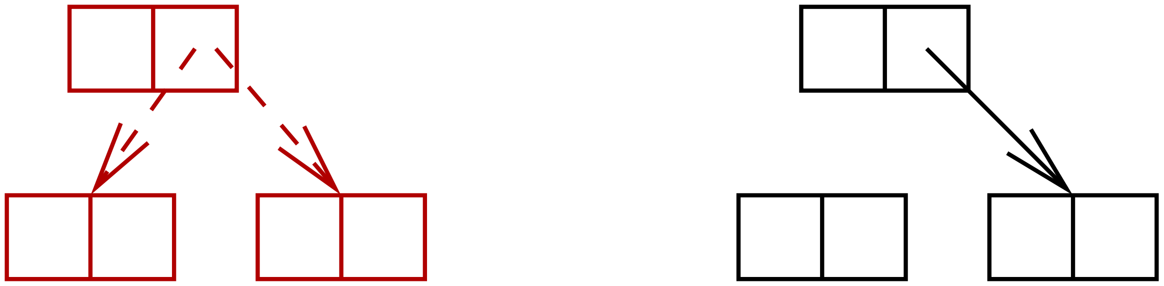
Typical requirement: $c \in \gamma(a)$ implies $\mathcal{I}_C(c) \supseteq \mathcal{I}_A(a)$
or $c \in \gamma(a)$ implies $\mathcal{I}_C(c) \subseteq \mathcal{I}_A(a)$ (more to say)

What does an abstract transition denote?

what may possibly execute:



what may possibly be pointed:



What does “dashed” denote? *a0 - - -> a1*

Overapproximation — $1/2$ — may/possibly: The corresponding concrete structure may or may not possess this transition, but all concrete transitions are “covered” by transitions of this form.

In operational semantics and process algebra, this is formalized as a simulation:

Given $\gamma : A \to \mathcal{P}(C)$, $K_C = \langle C, \tau_C, \mathcal{I}_C \rangle$, $K_A = \langle A, \tau_A, \mathcal{I}_A \rangle$, K_C is γ-simulated by K_A (written $K_C \lhd_\gamma K_A$)

iff for all $a \in A$, $c \in \gamma(a)$, $c' \in C$,

1. $\mathcal{I}_C(c) \subseteq \mathcal{I}_A(a)$
2. $c \to c'$ implies there exists $a' \in A$ such that $c' \in \gamma(a')$ and $a -- \to a'$.

That is, K_A “mimicks” the transitions and atomic properties of K_C.

Example over-approximation: $A = \{\bot, a0, a12, \top\}$

$$\alpha\{\} = \bot$$

$$\alpha\{c0\} = a0$$

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

$\alpha S = \top,$ otherwise

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

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

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

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

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

$$\begin{tikzcd}
\text{if } c \arrow[d, "\text{and}"'] & \in & \gamma(a) \arrow[d, "\text{then}"'] \\
c' & \in & \gamma(a')
\end{tikzcd}$$

$$\begin{tikzcd}
 & c1 \arrow[loop above] \arrow[dr, bend left] & \\
c0 \arrow[ur, bend left] \arrow[rr, bend right] & & c2
\end{tikzcd}$$

$$\begin{tikzcd}
\top \arrow[dr, dashed] & \\
a0 \arrow[r, dashed] & a12 \arrow[loop above, dashed] \\
\bot &
\end{tikzcd}$$

De£ning the abstract transition relation, τ_A, so that it gives a simulation

Recall the recipe for de£ning a functional transition:

$op_A = \alpha \circ op_C^+ \circ \gamma$

(Note: op_C^+ lifts op_C to compute on sets.)

But here we have a relation, τ_C — not a function — to approximate. Dams noted that, τ_A, the minimal overapproximation of τ_C goes

$a \dashrightarrow \alpha(s')$ iff $s \in \gamma(a)$ and $s \longrightarrow s'$.

The challenge lies in £nitely computing $\tau_A.a$, that is, the image of a in τ_A.

What properties can we safely check?

Aliasing — a is possibly tail-aliased:

$$\mathit{isAliased}(a) = \exists x.\exists y.\tau_{tail}(x,a) \wedge \tau_{tail}(y,a) \wedge x \neq y$$

$$a \models (\exists \tau^{-1}_{tail}.\mathrm{at}\ x) \wedge (\exists \tau^{-1}_{tail}.\mathrm{at}\ y)$$

(recall $a \models \exists R.\phi$ iff exists a' such that $R(a, a')$ and $a' \models \phi$)

$$\mathit{isAliased}(a) = \exists x.\exists y.(x \mapsto \_, a) * (y \mapsto \_, a) * \mathrm{true}$$

that is, $\exists x.\exists y.\tau_{tail}(x,a) * \tau_{tail}(y,a) * \mathrm{true}$

Reachability — a is possibly reachable from x:

$$r_x(a) = \tau^{*}_{tail}(x, a)$$

$$a \models \mu Z.\mathrm{at}\ x \vee \exists \tau^{-1}_{tail}.Z$$

$$r_x(a) =^{\mathit{lfp}} (x = a) \vee (\exists a'.\tau_{tail}(x, a') * r_{a'}(a))$$

These are branching-time properties.

Reachability — necessarily, all nodes reached from a are "happy":

$$\mathit{Happy}(a) = \forall y.\tau^*_{tail}(a,y) \supset \mathit{happy} \in \mathcal{I}_A(y)$$

$$a \models \nu Z.\mathit{isHappy} \wedge \forall\tau_{tail}.Z$$

(Assumes that $\mathcal{I}_A(a) \subseteq \mathcal{I}_C(c)$, when $c \in \gamma(a)$.)

That is, there does not exist a reachable node/cell that lacks happy.

End cell — necessarily, there is no cell linked to a:

$$\mathit{noTail}(a) = \forall y.\neg\tau_{tail}(a,y)$$

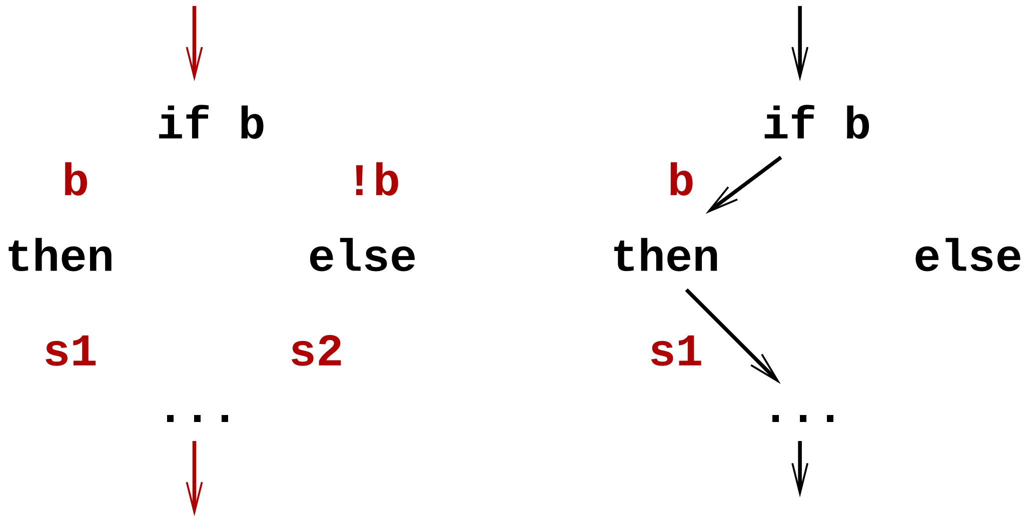
$$a \models \forall\tau_{tail}.\mathit{false}$$

That is, there does not exist a tail-transition from a.

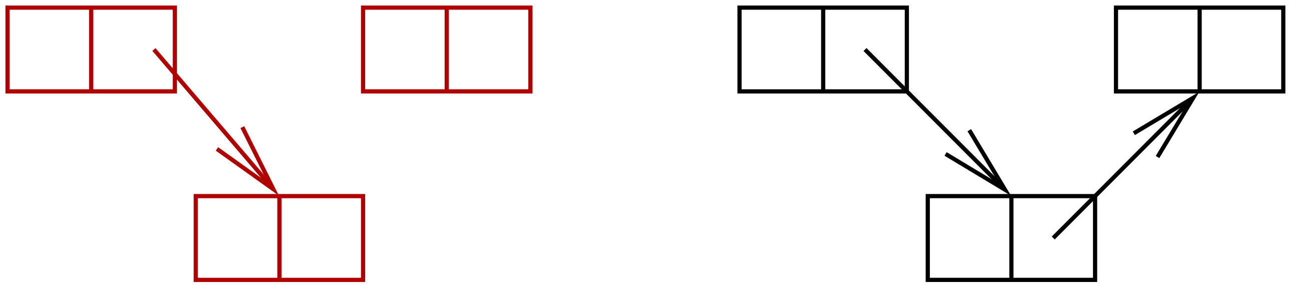
With the over-approximation model, we validate "universal properties" (by invalidating existential ones).

What does an abstract transition denote (2)?

what must necessarily execute:



what must necessarily be linked:



What does “solid” denote? $\boldsymbol{a0} \longrightarrow \boldsymbol{a1}$

Underapproximation — 1 — must/necessarily: All corresponding concrete structures must possess this transition.

This is a (dual) simulation:

Given $\gamma : A \to \mathcal{P}(C)$, $K_C = \langle C, \tau_C, \mathcal{I}_C \rangle$, $K_A = \langle A, \tau_A, \mathcal{I}_A \rangle$, K_A is dual-γ-simulated by K_C (written $K_A \lhd_\gamma^{-1} K_C$)

iff for all $a \in A$, $c \in \gamma(a)$, $a' in A$,

1. $\mathcal{I}_C(c) \supseteq \mathcal{I}_A(a)$
2. $a \longrightarrow a'$ implies there exists $c' \in C$ such that $c' \in \gamma(a')$ and $c \longrightarrow c'$.

That is, K_C “mimicks” the must-transitions and atomic must-properties of K_A.

Example under-approximation: $A = \{\bot, \mathrm{a0}, \mathrm{a12}, \top\}$

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

$$\alpha\{\} = \bot \qquad \gamma(\mathrm{a0}) = \{\mathrm{c0}\}$$

$$\alpha\{\mathrm{c0}\} = \mathrm{a0} \qquad \gamma(\mathrm{a12}) = \{\mathrm{c1}, \mathrm{c2}\}$$

$$\alpha\{\mathrm{c1}\} = \mathrm{a12} = \alpha\{\mathrm{c2}\} \qquad \gamma(\top) = \{\mathrm{c0}, \mathrm{c1}, \mathrm{c2}\}$$

$\alpha S = \top$, otherwise

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

$$\begin{tikzcd}
& \bot \arrow[r] \arrow[dl, bend right] \arrow[dr] & \top \\
\textit{a0} \arrow[rr] & & \textit{a12}
\end{tikzcd}
\qquad
\begin{tikzcd}
& \textit{c1} \arrow[loop above] \arrow[d, bend left] \\
\textit{c0} \arrow[ur, bend left] \arrow[r, bend right] & \textit{c2}
\end{tikzcd}$$

What properties can we safely check?

a is necessarily reachable from x:

$$r_x(a) = \tau^*_{tail}(x, a)$$

$$a \models \mu Z.\mathit{at}\ x \vee \exists \tau^{-1}_{tail}.Z$$

$$r_x(a) =^{lfp} (x = a) \vee (\exists a'.\tau_{tail}(x, a') * r_{a'}(a))$$

possibly, all cells reached from a are “safe”:

$$\mathit{isSafe}(a) = \forall y.\tau^*_{tail}(a, y) \supset \mathit{happy} \in \mathcal{I}_A(y)$$

$$a \models \nu Z.\mathit{isHappy} \wedge \forall \tau_{tail}.Z$$

(Assumes that $\mathcal{I}_A(a) \supseteq \mathcal{I}_C(c)$, when $c \in \gamma(a)$.)

That is, there does not exist a necessarily-reachable cell/node that lacks happy — the possibility that all reachable cells are happy still exists.

With an under-approximation model, we validate “existential properties” (and refute universal ones).

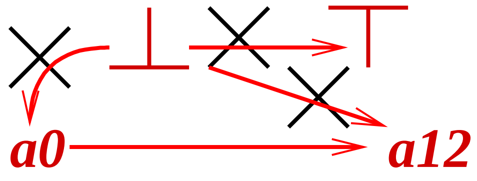
Mixed and modal transition systems

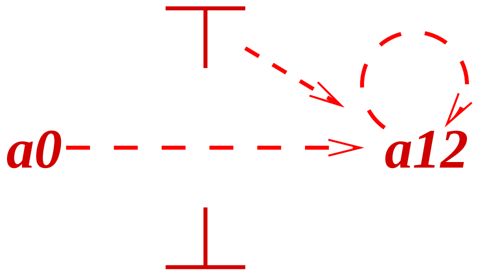
A mixed Kripke transition system is two systems, an under approximation and an over approximation, with the same node/cell/state set:

$$\langle \Sigma, \tau^{\mathrm{must}}, \tau^{\mathrm{may}}, \mathcal{I}^{\mathrm{must}}, \mathcal{I}^{\mathrm{may}} \rangle$$

When $\tau^{\mathrm{must}} \subseteq \tau^{\mathrm{may}}$ and $\mathcal{I}^{\mathrm{must}} \sqsubseteq \mathcal{I}^{\mathrm{may}}$, the system is modal.

When $\tau^{\mathrm{must}} = \tau^{\mathrm{may}}$ and $\mathcal{I}^{\mathrm{must}} = \mathcal{I}^{\mathrm{may}}$, the system is concrete — an ordinary Kripke transition system.





Simulation is replaced by re£nement:

Given $M_C = \langle C, \tau_C^{must}, \tau_C^{may}, \mathcal{I}_C^{must}, \mathcal{I}_C^{may} \rangle$ and $M_A = \langle A, \tau_A^{must}, \tau_A^{may}, \mathcal{I}_A^{must}, \mathcal{I}_A^{may} \rangle$,

M_C re£nes M_A iff

$$\langle C, \tau_C^{may}, \mathcal{I}_C^{may} \rangle \lhd_\gamma \langle A, \tau_A^{may}, \mathcal{I}_A^{may} \rangle$$

and

$$\langle A, \tau_A^{must}, \mathcal{I}_A^{must} \rangle \lhd_\gamma^{-1} \langle C, \tau_C^{must}, \mathcal{I}_C^{must} \rangle$$

That is, M_A's may-parts simulate M_C's, and M_C's must-parts dual-simulate M_A's.

When M_C re£nes M_A,

- M_C's under-approximation is larger (more precise) than M_A's
- M_C's over-approximation is smaller (more precise) than M_A's.

When M is concrete, its under and over-approximations coincide; they are exact.

We can validate a full predicate logic on a MTS

We validate universal subformulae on the upper-approximation and existential subformulae on the lower-approximation, jumping “back and forth” as needed.

We validate a negated formula by refuting it on the dual approximation.

Example:

$$\begin{tikzcd}
 & \top \arrow[rd, dashed] & \\
a0 \arrow[rr] \arrow[rr, dashed, shift right] & & a12 \arrow[loop above, dashed]
\end{tikzcd}$$

$$a0 \models^{under} \exists\tau.\forall\tau.\neg at\_a0$$

iff $a12 \models^{over} \forall\tau.\neg at\_a0$

iff $a12 \models^{over} \neg at\_a0$

iff $a12 \not\models^{under} at\_a0$

iff $true$

For a MTS, where $\tau_{must} \subseteq \tau_{may}$, there are only three possible outcomes: ϕ necessarily holds, ϕ possibly holds, ϕ not possibly holds.

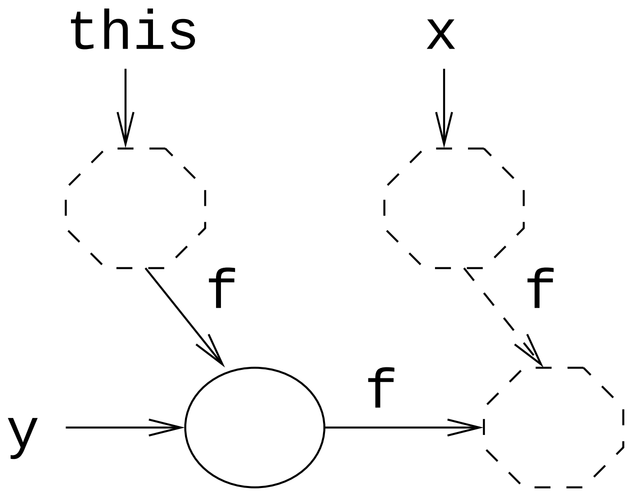
TVLA models have must-may nodes/cells such that *(i)* a must-node can not be split (or merged) in a re£nement; *(ii)* a may-node can not be merged in a re£nement. We might de£ne an extension of MTS with such nodes. (We might also restrict γ!)

The re£nement relation, quotiented, is a partial ordering in a dcpo of modal transition systems. Given MTS, M, its re£nements form a Kripke model unto which we can apply a modal logic:

- $M \models \Box\phi$ — all re£nements satisfy ϕ (intuitionistic)
- $M \models \Box\Diamond\phi$ — always possible to re£ne to satisfy (dense)
- Generalized model checking examines only the limit points of M's Kripke model. (Aprés Michael, these coincide.)

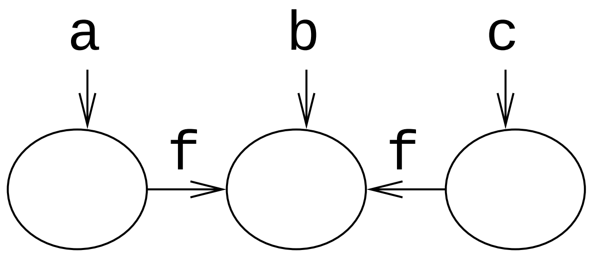
Whaley and Rinard's representation

```
class C {
  private C f;
  public void m(C x) {
    C y = new C();
    this.f = y;
    y.f = x.f; }
}
```



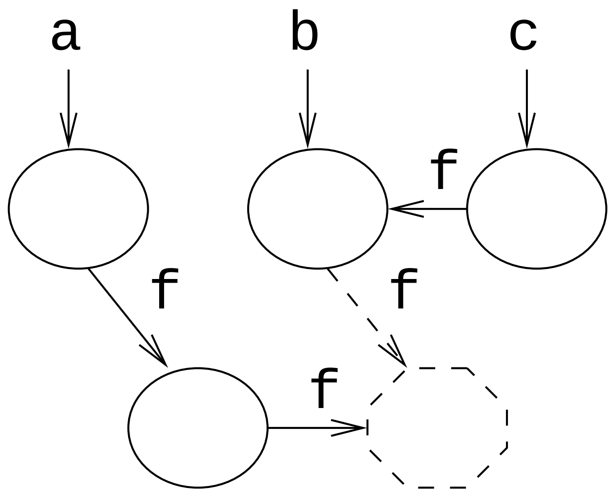
An invocation, like a.m(b), links the actual's shape to the formal's:

Caller context:



(perhaps we know nothing of b's links)

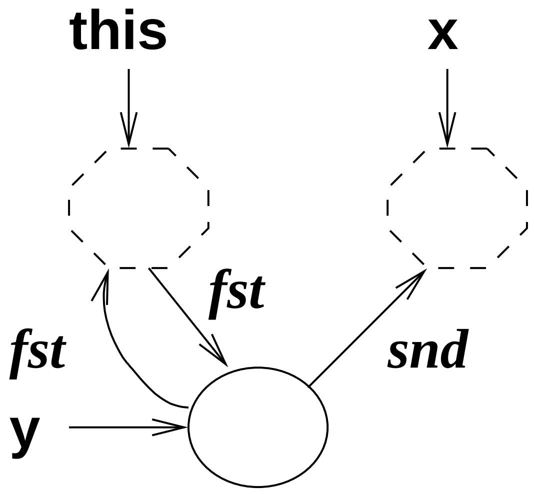
Result of binding:



Application of separation logic to Whaley-Rinard analysis

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

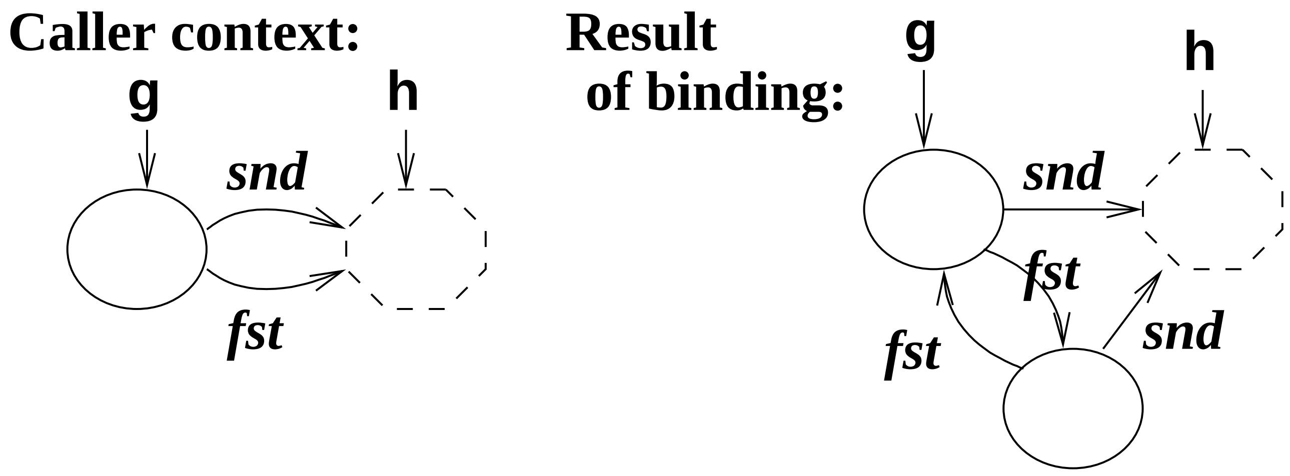
```
class C {
   C fst; C snd;
   function m(C x)  {
      C y := new C(this,x);
      this.fst := y; }
}
```



The shape analysis is neatly summarized as this Hoare triple:

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

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



The corresponding step in separation logic is

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

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

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

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

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

Using separation logic as an abstract semantic domain

There are precedents: CousotCousot79 used predicate logic assertions as an abstract computation domain and used the sp and wp rules as the abstract program operations for a `while`-language.

Because sp and wp are sound and relatively complete, the transformers, used as abstract operations on the assertions as input data, do not lose precision.

That is, the transformers were complete with respect to the Galois connection that mapped assertions to the store sets they denoted:

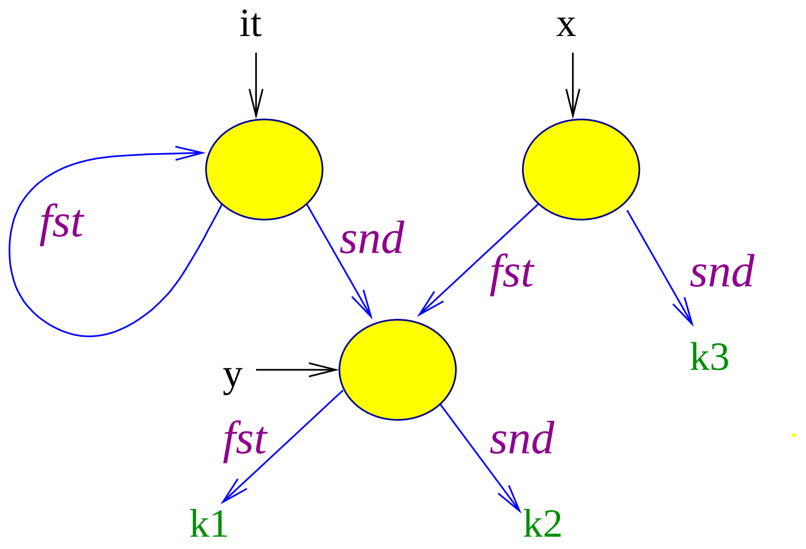
Forwards completeness: $\alpha(\mathrm{op}_C(S)) = \mathrm{op}_A(\alpha(S))$, where

$$\mathrm{op}_A = \alpha \circ \mathrm{op}_C \circ \gamma.$$

If we start with an initial S_0 in the image of γ, then all subsequent abstract computation steps remain in the image of γ.

“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,” $\mathtt{it}$:

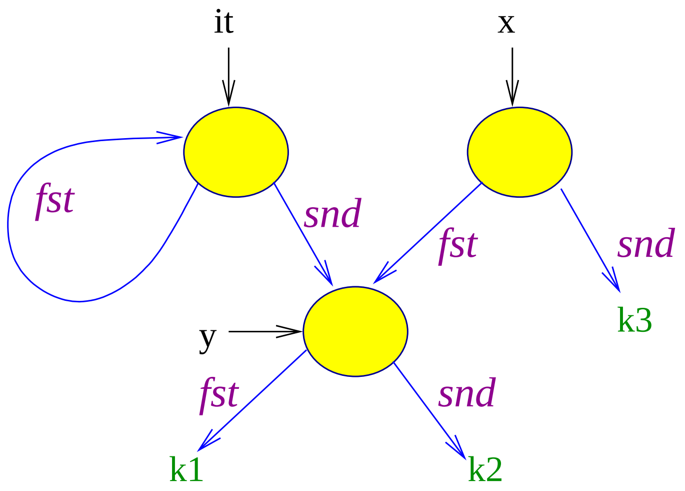
$$\{fst^i \mid i \geq 0\} \qquad \{fst^i.snd \mid i \geq 0\}$$

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

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:



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

The paths have been normalized by the cancellation law,

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

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 `x` to it, the result/heap-entry.

In LTL, traces are assumed to be in£nite, so we can extend each such (£nite) π by suf£xing $(\mathsf{done}^{-1})^{\omega}$ to it.

We can ask standard questions:

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

When the objects are ML-typed, Blanchet used the ML-types as fsa’s and represented the in£nite-cardinality trace sets as a £nite set of fsa-state names.

A model check can be performed on the state names by using the ML-type-fsa’s as the £nite-state structures that are linear-time model checked.

I do not know of a serious development of linear-time shape analysis, but it makes good sense to try it!

Paths semantics of the pairs language

A expression evaluates to a $\mathsf{Pathset}$:

$$\mathsf{Pathset} = \mathcal{P}(\mathsf{Path})$$

$$\mathsf{Path} = \mathsf{Const}.\mathsf{Selector}^*$$

$$\mathsf{Selector} = \{\mathit{fst}^{-1}, \mathsf{snd}^{-1}, \mathit{fst}, \mathsf{snd}\}$$

A path travels from a point of interest (here, the constants) to the result of the expression ("it"):

$$\vdash k \Downarrow \{k\} \qquad \frac{\vdash e \Downarrow S}{\vdash e.\mathit{fst} \Downarrow S \circ \mathit{fst}}$$

$$\frac{\vdash e_i \Downarrow S_i \quad i \in 1..2}{\vdash (e_1, e_2) \Downarrow S_1 \circ \mathit{fst}^{-1} \ \cup \ S_2 \circ \mathit{snd}^{-1}}$$

Note: $\circ$ is path composition, $S \circ i = \{s \cdot i \mid s \in S\}$,

$i \in \{\mathit{fst}, \mathit{snd}, \mathit{fst}^{-1}, \mathit{snd}^{-1}\}$, where destructors cancel constructors: $p \cdot \mathit{fst}^{-1} \cdot \mathit{fst} = p$.

Inserting regions into the paths semantics

Let assertions have the format, $S_{x_1} * S_{x_2} * \cdots * S_{x_n}$, where each S_{x_i} has form, $\{v.s^* = x_i\}$, describing paths from values, v, to the region's entry point, x_i. (The $*$ asserts that the paths in region S_x are use objects that are disjoint from all other heap regions, S_y, $Y \neq x$.)

$$\vdash k \Downarrow \{k = it\}$$

$$\frac{\vdash e_1 \Downarrow S_1 \qquad \vdash e_2 \Downarrow S_2}{\vdash (e_1, e_2) \Downarrow \{m.\mathit{fst}^{-1} = it,\ \ n.\mathit{snd}^{-1} = it\} \ * \ [m/it]S_1 \ * \ [n/it]S_2}$$

(Note: m and n are implicitly existentially quanti£ed.)

$$\frac{\vdash e \Downarrow S}{\vdash e.\mathit{fst} \Downarrow S \circ \mathit{fst}} \quad \text{where} \quad \begin{aligned} &(S_{x_1} * S_{x_2} * \cdots * S_{x_n} * S_{it}) \circ i \\ &= S_{x_1} * S_{x_2} * \cdots * S_{x_n} * (S_{it} \circ i) \end{aligned}$$

Summary

- For analyses that deduce properties of paths, under- and over-approximation issues are crucial.
- Branching-time models of heap are widely used, but maybe Deutsch and Blanchet know better — end-users prefer linear-time logic over branching time; shouldn't we?
- Integration of spatial logics into heap abstraction and static analysis seems worth a try.

References

1. This talk: `www.cis.ksu.edu/~schmidt/papers`
Will be posted next week, with the following bib list completed:
2. Bruno Blanchet. Escape analysis: application to ML and Java. PhD thesis, Ecole Polytechnique, Paris, 2000.
3. Patrick and Radhia Cousot. Systematic design of program analysis frameworks. POPL 1979
4. D. Dams, R. Gerth, O. Grumberg. Abstract interpretation of reactive systems. ACM TOPLAS (19) 1997.
5. Alain Deutsch. Operational semantics of programming languages and representation in relations on rational languages. PhD thesis, Univ. of Paris VI, 1992.
6. M. Huth, R. Jagadeesan, D. Schmidt. A domain equation for re£nement of partial systems. MSCS, in print.

7. Kim Larsen. Modal speci£cations. CAV’89, LNCS 407.

8. M. Sagiv, T. Reps, R. Wilhelm. Parametric shape analysis via 3-valued logic. POPL 1999.

9. Moshe Vardi. Branching vs. linear time: £nal showdown. Invited talk, ETAPS 2001.