Generalized model checking for shape analysis?

Michael Huth
Computing
Imperial College London

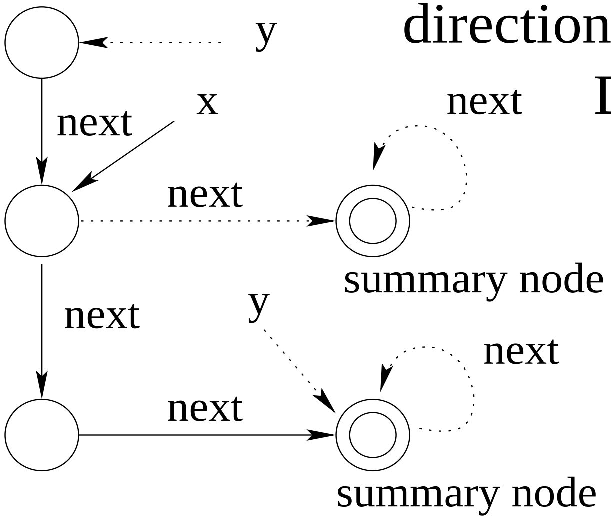
Collaborators: Patrice Godefroid, Rahda Jagadeesan, Shekhar Pradhan, and David Schmidt.

Objectives and outline of talk

1. give brief primer on compositional model checking of modal shape graphs (MSGs);
2. discuss generalized model checking (GMC) for MSGs;
3. define and motivate *robust* frameworks for generalized model checking;
4. prove that MSGs with 'true' shape theory are robust; and
5. sketch which type constructors preserve robustness of operational/denotational models.

1. Compositional checks of MSGs . . .

An example check



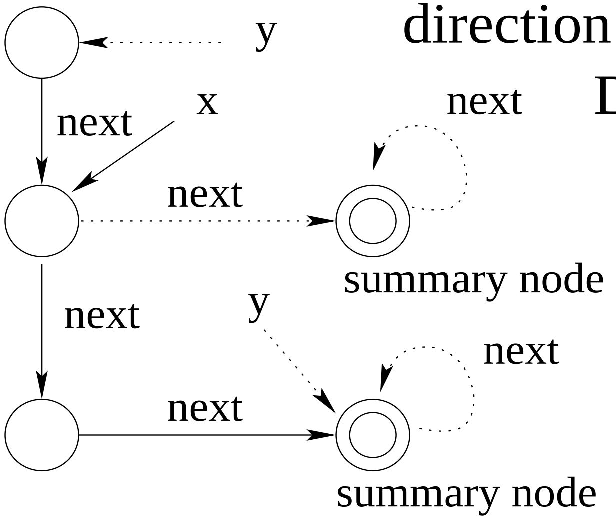
direction relationship (Ghiya & Hendren'96):

D(x,y) = EF _next [x & EF _next (y)]

$M \not\models^{\mathrm{a}} \neg \mathsf{EF}_{\mathtt{next}}(y) \vee D(x,y) \vee D(y,x)$ — where $\models^{\mathrm{a}}$ is assertion mode — since $M \models^{\mathrm{c}} \mathsf{EF}_{\mathtt{next}}(y) \wedge \neg D(x,y) \wedge \neg D(y,x)$ — where $\models^{\mathrm{c}}$ is consistency mode (each conjunct is consistent!).

2. Generalized model checks of MSGs . . .

The example check re-visited

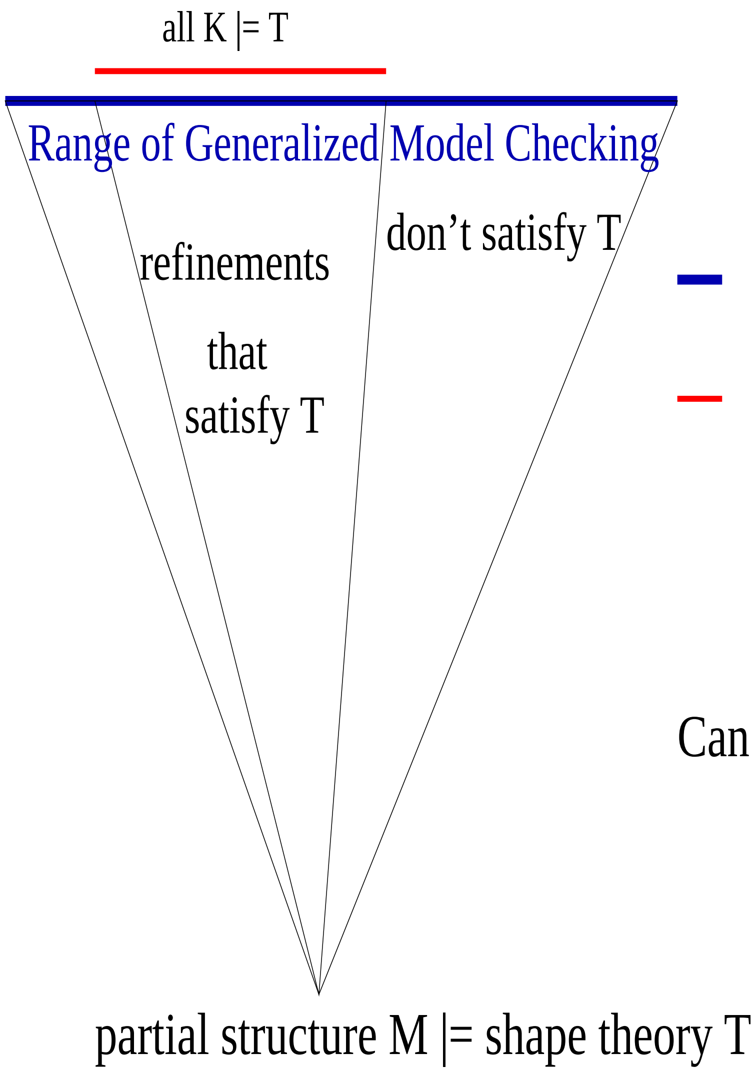


direction relationship (Ghiya & Hendren'96):

D(x,y) = EF _next [x & EF _next (y)]

We showed $M \not\models^{\mathrm{a}} \neg \mathsf{EF}_{\mathtt{next}}(y) \vee D(x,y) \vee D(y,x)$, but $M \models_{\mathrm{t}} \neg \mathsf{EF}_{\mathtt{next}}(y) \vee D(x,y) \vee D(y,x)$ holds — where $\models_{\mathrm{t}}$ thorough semantics (Bruns & Godefroid'00): $M \models_{\mathrm{t}} \phi$ iff all 'implementations' of M satisfy ϕ. Hence, $\models^{\mathrm{a}}$ under-approximates and $\models^{\mathrm{c}}$ over-approximates.

GMC for a shape theory 1



all ‘implementations’

all ‘implementations’ of right shape

Can GMC be parametric in shape theory?

GMC for a shape theory 2

GMC *without* shape: sound for GMC for *any* shape theory.

Current CTL GMC (Bruns & Godefroid'00):

- M a pointed modal transition system (MTS; Larsen & Thomsen'88), ϕ CTL formula;
- construct alternating Büchi word automata $\mathcal{A}_{(M,\phi)}$ over 1-letter alphabet, $\leq O(|\, M \,| \cdot 2^{O(|\phi|)})$ states;
- its language non-empty iff $\exists$ *concrete* pointed MTS K with $M\ \mathsf{ref'd\_by}\ K$, $K \models \phi$.

Can automata-theoretic techniques be adapted to deal with shape?

Which shape theories give rise to regular languages?

GMC for shape theory 3

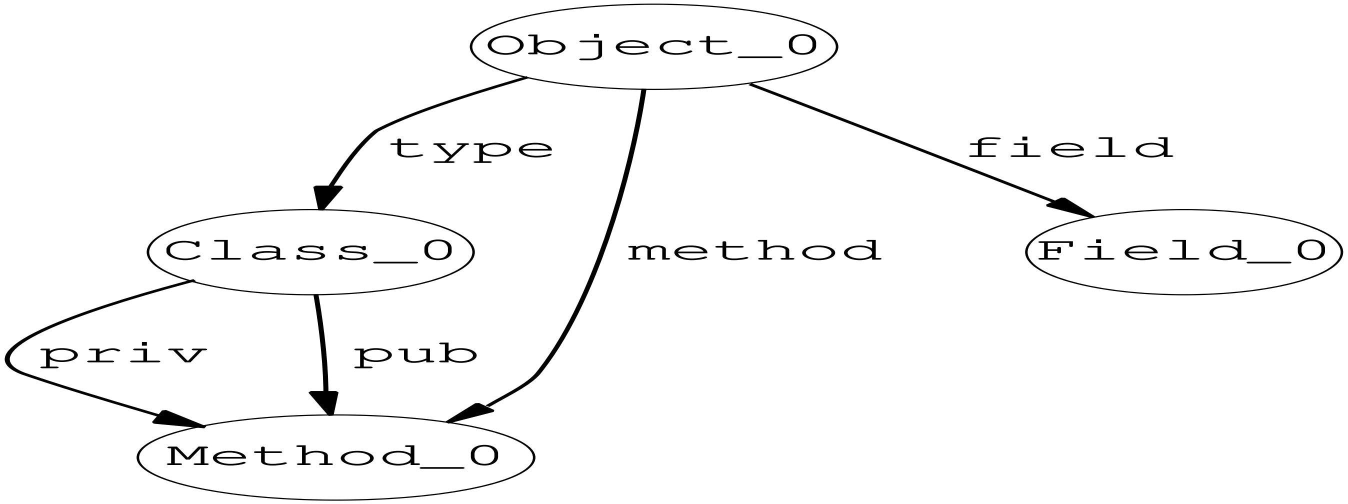
Daniel Jackson's Alloy constraint analyzer:

- modular language for finding models of NP formulas (Tom's Wed talk);
- specify only what matters; rest is *implicitly* '3-valued;'
- requires bounds on set domains;
- abstraction of arithmetic etc;
- suitable for 'proofs' (Greta's talk) only if bounds are general enough;

One-minute course on Alloy

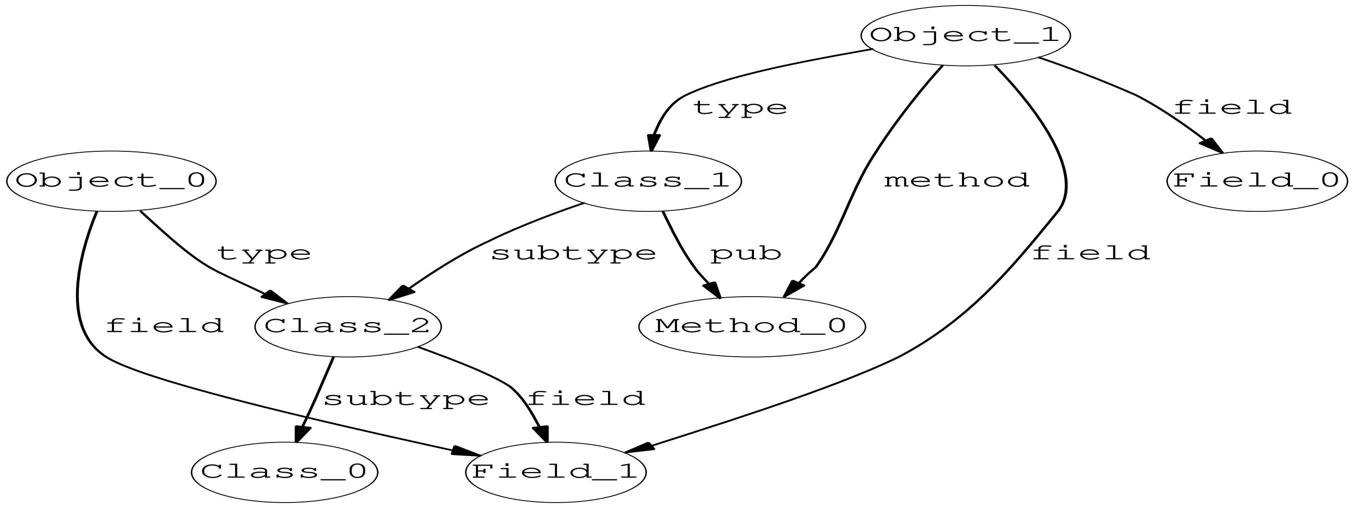
```
sig Class { pub, priv: set Method,
            field: set Field,
            subtype: set Class }

sig Object { type: Class,
             method: set Method,
             field: set Field }

fact { all o: Object | o.method = o.type.(pub + priv) }
-- consistency check ``fun''
fun Scenario(Object_1,Object_0: Object) { some Object_1.method
 Object_1.type != Object_0.type
 Object_1.type.^subtype = Class - Object_1.type

assert Safety{  -- assertion check
 all o: Object | no o.method & o.type.pub & o.type.priv }
```

Counterexample to assertion `Safety`



- state space bounded in size of signatures — bounds user-defined;
- abstract model: stated only what mattered to us; explored consequences of this partiality;
- state space: doubly exponential in size of signatures $\Rightarrow$ depends on *small scope hypothesis*;
- back-end tool: SAT solvers.

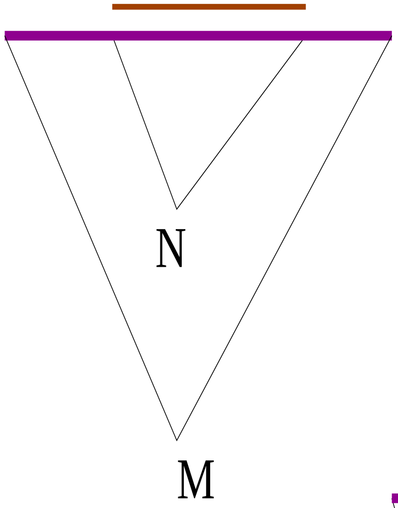
A simulation of `Scenario`



- one object has some method(s), its type has all other classes as (transitive) subtypes; and another object is of a different type;
- may add constraints in `Scenario` to "guide" simulation;
- partiality beyond MTSs: no fixed state space; e.g. no fixed communication topology; and
- $\Rightarrow$ parametric specifications of protocols/algorithms/designs.

3. Robust generalized model checking . . .

Robustness

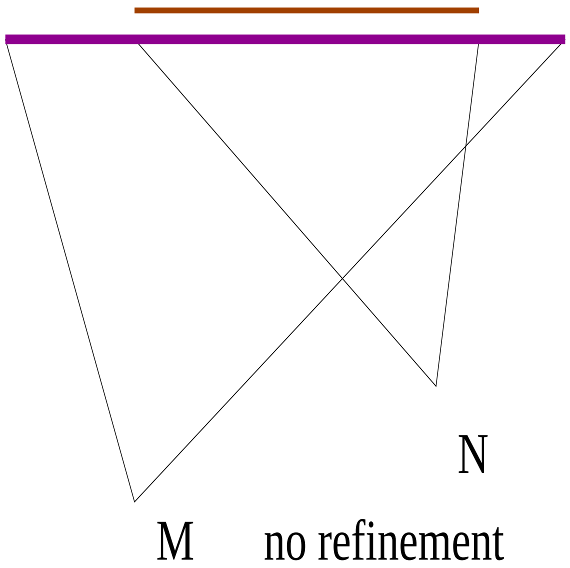


co−inductive

M ref'd_by N implies

'implementations' of N also 'implementations' of N

Converse could be false:



no refinement

Conceptually ...

- refinements define implementations; any refinement preorder $\mathsf{ref'd\_by}$ between shape graphs (over shape theory T) determines implementation relation: for $M \models T$, $M\ \mathsf{impl'd\_by}\ K$ iff
  1. $M\ \mathsf{ref'd\_by}\ K$,
  2. $\forall K' : K\ \mathsf{ref'd\_by}\ K' \Rightarrow K'\ \mathsf{ref'd\_by}\ K$, and
  3. $K \models T$;
- conversely, any $\mathsf{impl'd\_by}$ determines some $M\ \mathsf{ref'd\_by}\ N$ defined by $\{K \mid N\ \mathsf{impl'd\_by}\ K\} \subseteq \{K \mid M\ \mathsf{impl'd\_by}\ K\}$;
- robustness $=$
  - this inclusion *characterizes* refinement; and
  - implementation and refinement relation are definable in terms of each other, render same concept.

A shape theory for lists

- Each local variable has at most one outbound solid arc;
- each object has at most one outbound solid next-arc.

Expressible through multiplicity constraints on model signature;
e.g. for variables

- $L^{\mathrm{a}}\colon \mathrm{Var} \rightarrow ?\ \Sigma$;
- $L^{\mathrm{c}}\colon \mathrm{Var} \rightarrow \Sigma$.

Consistency requires: for all x, $L^{\mathrm{a}}(x) \subseteq L^{\mathrm{c}}(x)$.

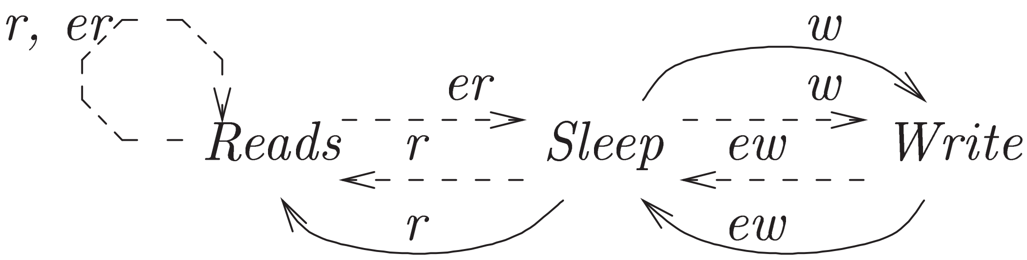
Up to co-induction, the order is $({L_1}^{\mathrm{a}}, {L_1}^{\mathrm{c}}) \leq ({L_2}^{\mathrm{a}}, {L_2}^{\mathrm{c}})$ iff for all x,
${L_1}^{\mathrm{a}}(x) \subseteq {L_2}^{\mathrm{a}}(x)$ and ${L_2}^{\mathrm{c}}(x) \subseteq {L_1}^{\mathrm{c}}(x)$.

'Implementations:' $L^{\mathrm{a}} = L^{\mathrm{c}}$.

4. Robustness for ‘true’ shape theory . . .

A modal transition system

…has two kinds of transitions: R^{a} (solid; **must** implement) and R^{c} (dashed; **may** implement) with $R^{\mathrm{a}} \subseteq R^{\mathrm{c}}$, e.g. (HJS'03)



Let $\models^{\mathrm{a}}$ and $\models^{\mathrm{c}}$ abstract "for all implementations" and "for some implementation" (respectively).

- $(M, i) \models^{\mathrm{a}} \neg(\exists w)(\exists r)\mathsf{true}$ holds at all i, because $(M, i) \models^{\mathrm{c}} (\exists w)(\exists r)\mathsf{true}$ doesn't — there is no sequence of two R^{c}-transitions labeled by w and then r.
- $(M, \mathit{Sleep}) \models^{\mathrm{c}} (\exists r)(\exists r)(\exists er)(\exists w)\mathsf{true}$ holds, because we can find (at Sleep) a sequence of these four R^{c}-transitions in order.

Domain-theoretic model of MTSs (HJS'03)

- Initial solution $\mathcal{D}$ of

$$D = \prod_{\alpha \in \texttt{Act}} \mathsf{M}[D]$$

  in the category of SFP domains and Scott-continuous maps.

- $\mathsf{M}[D]$, mixed powerdomain of D (Heckmann'90, Gunter'92);
- elements of $\mathsf{M}[D]$: (L, U) where L Scott-closed, U Scott-compact upper, and $L = {\downarrow}(L \cap U)$; and
- the order on $\mathsf{M}[D]$ is $(L, U) \leq (L', U')$ iff $L \subseteq L'$ and $U' \subseteq U$.

Example elements:

- $\perp_{\mathcal{D}} = (\emptyset, \mathcal{D})_{\alpha \in \texttt{Act}}$ is the "universal stub" aka "state-wide divergence;" and
- $(\emptyset, \emptyset)_{\alpha \in \texttt{Act}}$ is "state-wide deadlock."

Adequacy of domain model

For all MTS M and N with designated start state, 1.-4. in (HJS'03),

1. there is an embedding $\langle\!| M |\!\rangle$ in $\mathcal{D}$;
2. M $\mathsf{ref'd\_by}$ $\langle\!| M |\!\rangle$ and $\langle\!| M |\!\rangle$ $\mathsf{ref'd\_by}$ M;
3. M $\mathsf{ref'd\_by}$ N iff $\langle\!| M |\!\rangle$ $\mathsf{ref'd\_by}$ $\langle\!| N |\!\rangle$;
4. on $\mathcal{D}$, $\mathsf{ref'd\_by}$ is equal to the order $\leq$ of the domain $\mathcal{D}$;
5. $L \in \mathrm{LTS} \Rightarrow \langle\!| L |\!\rangle \in \max(\mathcal{D})$; and
6. $d \in \max(\mathcal{D}) \Rightarrow d$ determines a 'topologically compact' LTS (which may not be image-finite in the discrete sense).

$\Rightarrow (\mathcal{D}, \leq)$ is a precise model of MTS-framework if we reason up to equivalence induced by $\mathsf{ref'd\_by}$ only.

Proof of robustness for MTSs

We write

$$d \leq_t e \stackrel{\text{def}}{=} \{m \in \max(\mathcal{D}) \mid e \leq m\} \subseteq \{m \in \max(\mathcal{D}) \mid d \leq m\}.$$

Adequacy of model $\Rightarrow$ suffices to show that $d \leq_t e \Rightarrow d \leq e$.

1. We show that $\max(\mathcal{D})$ is compact in D (uses techniques of (Alessi et al.'03);
2. Hofmann-Mislove theorem and 1. $\Rightarrow$ e above may be compact; (d may be anyway);
3. define "measure of partiality" $k \mapsto \deg(k) \colon \mathbf{K}(\mathcal{D}) \to \mathbb{N}$ with $\deg(k) = 0$ iff $k \in \max(\mathcal{D})$;
4. prove claim by induction on $\deg(e)$; base step = previous item; inductive step: e infimum of $e[+\alpha, l]$ and $e[-\alpha, l]$ with smaller degree, which do(+)/don't(-) implement $e \to^* l \to^\alpha$.

5. Robust type constructors . . .

Motivation

- Type constructors are building blocks of domain equations for adequate models of frameworks (e.g. $\prod$ and $\mathsf{M}[]$ for our $\mathcal{D}$);
- a domain $(D, \leq)$ is *robust* iff $\leq_t$ equals $\leq$;
- which type constructors (e.g. functors in SFP) preserve robustness?
- does such a preservation extend to the solution of recursive equations (e.g. inverse limits)?

Preliminary results

- the unary lift functor maps robust $\{*\}$ to non-robust $\{\perp < *\}$;

- the lower powerdomain enforces a top element;

- the upper powerdomain $\mathsf{U}[D]$: $X \leq_t Y$ iff $Y \cap \max(D) \subseteq X \cap \max(D)$;

\+ the Scott-continuous closure of robust domain;

? by previous item: the convex powerdomain is closed if the mixed powerdomain is;

? mixed powerdomain; topological conjecture: "yes, closed;"

\+ the product functors and coalesced sum functors;

- any sum functors which involves a lift; and

\+ Scott-continuous projections of robust domains, if maximal elements are projected to maximal elements in image.

Conclusions

(This page is intentionally left blank.)