Reasoning about sequences of memory states

R. Brochenin, S. Demri, E. Lozes

LSV, CNRS & ENS Cachan

Evolution Temporal Logic

Yahav, E., Reps, T., Sagiv, S., Wilhelm, R. : Verifying Temporal Heap Properties Specified via Evolution Logic. (ESOP 2003)

- **Syntax** :

$$\phi \ ::= \ 0 \mid 1 \mid p(v_1, .., v_n) \mid \odot v \mid \oslash v \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \exists v.\phi$$
$$\mid (TCv_1, v_2 : \phi_1)(v_3, v_4) \mid \phi_1 \mathrm{U} \phi_2 \mid \mathrm{X}\phi$$

- **Example formula : no memleak will occur**

$$\Box \forall v. \odot v \rightarrow \Diamond \oslash v$$

Each allocated cell (at some time) will deallocated

- **Models = sequences of "memory states"**

Navigation Temporal Logic

Dino Distefano, Joost-Pieter Katoen, Arend Rendsink
Who is pointing when to whom ? [FSTTCS'04]
Safety and Liveness in Concurrent Pointer Programs [FMCO'05]

- **Syntax**

$$\begin{array}{lll} \alpha & ::= & \texttt{null} \mid x \mid \alpha\uparrow \\ \phi & ::= & \alpha = \alpha \mid \alpha\mathsf{new} \mid \alpha \rightsquigarrow \alpha \mid \phi \wedge \phi \mid \neg\phi \mid \exists x.\phi \mid \mathsf{X}\phi \mid \phi\mathsf{U}\phi' \end{array}$$

- **Example formula : list reversal**

$$\forall x, y.((v \rightsquigarrow x \wedge x\uparrow = y) \Rightarrow \Diamond\Box(y\uparrow = x))$$

Do runs go too fast ?

What is the relation between two consecutive memory states ?

- Not clear in previous works : some restrictions to have a flavour of **concrete run**.
- e.g. : α new in NTL means α is allocated and was not before... ... but many changes are possible

Do runs go too fast ?

What is the relation between two consecutive memory states ?

- Not clear in previous works : some restrictions to have a flavour of **concrete run**.
- e.g. : α new in NTL means α is allocated and was not before...
  ... but many changes are possible
- not even a **big step semantics** :
  two consecutive states are not necessarily related by a finite run.

Do runs go too fast ?

What is the relation between two consecutive memory states ?

- Not clear in previous works : some restrictions to have a flavour of **concrete run**.
- e.g. : α new in NTL means α is allocated and was not before...
  ... but many changes are possible
- not even a **big step semantics** :
  two consecutive states are not necessarily related by a finite run.
- **What should be considered ?**
  Here we will consider :
  - arbitrary runs
  - concrete runs (from programs)
  - in between : runs with constant heap.

Temporal properties of pointer arithmetic

- **Example : block preservation.**

$$\Box \bigwedge_{i=0..n-1} \mathrm{x} + i \mapsto - \wedge \neg \mathrm{x} + n \mapsto -$$

- **Example : block scanning.**

$$\Box \mathrm{X}\mathrm{x} = \mathrm{x} \wedge \left(\bigvee_{i=0..n-1} \mathrm{x} + i \mapsto \mathrm{y} \;\; \mathrm{U} \mathrm{x} + n - 1 \mapsto \mathrm{y} \right)$$

- Limitations : $\mathrm{X}\mathrm{x} = \mathrm{x} + i$

Talking about recursion via LTL

- **A language for recursive data structures :**

$$\mathit{List}(\mathtt{x}) \stackrel{\mu}{=} \mathtt{x} \mapsto \{\mathit{next} : \mathtt{null}\} \vee \exists \mathtt{y}.\mathtt{x} \mapsto \{\mathit{next} : \mathtt{y}\} \wedge \mathit{List}(\mathtt{y})$$

- General recursion raises undecidability
  $\Rightarrow$ "LTL style" recursion :

$$\mathtt{x} \mapsto \{\mathit{next} : \mathsf{X}\mathtt{x}\} \ \mathsf{U} \ \mathtt{x} \mapsto \{\mathit{next} : \mathtt{null}\}$$

Here, "only variables are moving".

Talking about recursion via LTL

- **A language for recursive data structures :**

$$List(\mathtt{x}) \stackrel{\mu}{=} \mathtt{x} \mapsto \{next : \mathtt{null}\} \vee \exists \mathtt{y}.\mathtt{x} \mapsto \{next : \mathtt{y}\} \wedge List(\mathtt{y})$$

- General recursion raises undecidability
  $\Rightarrow$ "LTL style" recursion :

$$\mathtt{x} \mapsto \{next : \mathsf{X}\mathtt{x}\} \; \mathsf{U} \; \mathtt{x} \mapsto \{next : \mathtt{null}\}$$

  Here, "only variables are moving".

- Other formulas

$$\mathtt{x} = \mathtt{null} \; \wedge \; \left(\; (\mathsf{X}\mathtt{x}) \mapsto \{prev : \mathtt{x}; next : \mathsf{X}^2\mathtt{x}\} \; \mathsf{U} \; \mathsf{X}\mathtt{x} = \mathtt{null} \; \right)$$

  Trees ? (maybe requires CTL ?)

Programs as formulas

- Programs without update :
  ex : P =
  while $\mathtt{x} <> \mathtt{null}$ **do** $\mathtt{x} = \mathtt{x} \rightarrow \mathit{next}; \mathtt{y} = \mathtt{y} \rightarrow \mathit{next}$ **end**

  $$\phi_P \;=\; \big(\mathtt{x} \mapsto \{\mathit{next} : \mathrm{X}\mathtt{x}\} \;\wedge\; \mathtt{y} \mapsto \{\mathit{next} : \mathrm{X}\mathtt{y}\}\ \mathrm{U}\ \mathtt{x} = \mathtt{null}\big)$$

  More generally : ϕ_P for P without update.

- Describing the input/output relation with $\mapsto_0$ and $\mapsto_1$
  ex : list reversal

  $$\big(\mathtt{x} \mapsto_1 \{\mathit{next} = \mathrm{X}\mathtt{x}\} \;\wedge\; (\mathrm{X}\mathtt{x} \mapsto_2 \{\mathit{next} : \mathtt{x}\}\ \mathrm{U}\ \mathtt{x} = \mathtt{null}\big)$$

  More generally : single-pass programs ?

Heaps and models

A *memory state* is a pair (s,h) of :

- a *store* : s : $\mathtt{Var} \longrightarrow \mathbb{N}$
- a *heap* : h : $\mathbb{N} \rightharpoonup_{fin} (\mathtt{Lab} \rightharpoonup_{fin} \mathbb{N})$
  Intuition : dom h = allocated addresses.

A *model* is a sequence $(s_i, h_i)_{i<\alpha}$, finite or infinite, of memory states.

A *model with constant heap* is a sequence $(s_i, h)_{i<\alpha}$.

N.B. : $\mathtt{Mod}^{ct} \subset \mathtt{Mod}$.

The programming language

Syntax

$$
\begin{array}{rll}
\texttt{instr} ::= & \texttt{x} := \texttt{y} \mid \textit{skip} & \\
& \mid \texttt{x} := \texttt{y} \rightarrow l \mid \texttt{x} \rightarrow l := \texttt{y} & \text{(record programs)} \\
& \mid \texttt{x} := \texttt{cons}(l_1 : x_1, .., l_k : x_k) \mid \texttt{free x}, l & \\
& \mid \texttt{x} := \texttt{y}[i] \mid \texttt{x}[i] := \texttt{y} & \text{(array programs)} \\
& \mid \texttt{x} = \texttt{malloc}(i) \mid \texttt{free x}, i &
\end{array}
$$

Semantic

$$[\![P]\!](s_0, h_0) \; = \; \text{set of models } \textit{representing} \text{ executions}$$

N.B : If P has no destructive update, $[\![P]\!](s_0, h_0) \subset \texttt{Mod}^{ct}$.

The logic

- Expressions

$$e ::= \ \mathtt{x} \mid \mathtt{null} \mid \mathsf{X}e$$

- Atomic formulae

$$P ::= \ e = e' \mid \mathtt{x} + i \mapsto \{l : e\}$$

- State formulae

$$\begin{array}{lll} \mathcal{A} ::= & P & \\ & \mid \mathcal{A} * \mathcal{B} \mid \mathcal{A} \mathrel{-\!\!*} \mathcal{B} \mid \mathtt{emp} & \text{(spatial fragment)} \\ & \mid \mathcal{A} \wedge \mathcal{B} \mid \mathcal{A} \rightarrow \mathcal{A} \mid \top \mid \bot & \text{(classical fragment)} \end{array}$$

- Temporal formulae

$$\Phi ::= \ \mathcal{A} \mid \mathsf{X}\Phi \mid \Phi \mathsf{U} \Phi' \mid \Phi \wedge \Phi' \mid \neg\Phi$$

Semantics

$$
\begin{array}{llll}
s,h \models_{\mathrm{SL}} & e = e' & \text{iff } [e]_s = [e']_s, \text{ with } [\mathtt{x}]_s = s(\mathtt{x}) \text{ and } [\mathtt{null}]_s = nil. \\
s,h \models_{\mathrm{SL}} & \mathtt{x} + i \mapsto \{l : e\} & \text{iff } \mathtt{dom}(h) = \{s(\mathtt{x}) + i\} \text{ and } h(s(\mathtt{x}) + i) = [e]_s \\
s,h \models_{\mathrm{SL}} & \mathtt{emp} & \text{iff } \mathtt{dom}(h) = \emptyset \\
s,h \models_{\mathrm{SL}} & \mathcal{A}_1 * \mathcal{A}_2 & \text{iff } \exists\ h_1, h_2 \text{ s.t.} h = h_1 * h_2 \\
 & & \quad \text{and } \forall k \in \{1,2\}, s, h_k \models_{\mathrm{SL}} \mathcal{A}_k \\
s,h \models_{\mathrm{SL}} & \mathcal{A}' \mathrel{-\!\!*} \mathcal{A} & \text{iff } \forall h', \text{ if } h \perp h' \text{ and } s, h' \models_{\mathrm{SL}} \mathcal{A}' \\
 & & \quad \text{then } s, h * h' \models_{\mathrm{SL}} \mathcal{A}. \\
s,h \models_{\mathrm{SL}} & \mathcal{A}_1 \wedge \mathcal{A}_2 & \text{iff } \forall k \in \{1,2\}.\ s,h \models_{\mathrm{SL}} \mathcal{A}_k \\
s,h \models_{\mathrm{SL}} & \mathcal{A}' \rightarrow \mathcal{A} & \text{iff } s,h \models_{\mathrm{SL}} \mathcal{A}' \text{ implies } s,h \models_{\mathrm{SL}} \mathcal{A} \\
s,h \models_{\mathrm{SL}} & \perp & \text{never} \\
\rho, i \models & \mathsf{X}\Phi & \text{iff } i < |\rho| and \rho, i+1 \models \Phi. \\
\rho, i \models & \Phi \mathsf{U} \Phi' & \text{iff } \exists j \geq i, j \leq |\rho|,\ \rho, j \models \Phi', \\
 & & \quad \text{and } \forall k,\ i \leq k < j,\ \rho, k \models \Phi. \\
\rho, i \models & \mathcal{A} & \text{iff } s,h \models_{\mathrm{SL}} \mathcal{A}[\mathsf{X}^i\mathtt{x} \leftarrow \langle \mathtt{x}, i\rangle], \\
 & & \quad \text{where}: h = h_i \text{ and } s(\langle \mathtt{x}, k\rangle) = s_{i+k}(\mathtt{x}).
\end{array}
$$

The problems we considered

- *Satisfiability* ($\texttt{SAT}$, resp. $\texttt{SAT}^{ct}$) : given Φ of LTL_{mem}, is there $\rho \in \texttt{Mod}$ (resp. $\rho \in \texttt{Mod}^{ct}$) such that $\rho \models \Phi$?
- *Model checking* ($\texttt{MC}$, resp. $\texttt{MC}^{ct}$)) : given Φ of LTL_{mem}, a program $\texttt{p} \in \texttt{P}$ (resp. $\texttt{p} \in \texttt{P}^{ct}$), and a memory state (s_0, h_0), do $P, (s_0, h_0) \models \Phi$ holds ?
- *Program checking* ($\texttt{PC}$, resp. $\texttt{PC}^{ct}$) : given Φ of LTL_{mem} and a program $\texttt{p} \in \texttt{P}$ (resp. $\texttt{p} \in \texttt{P}^{ct}$), is there a memory state (s_0, h_0) such that $P, (s, h) \models \Phi$ holds ?

May express : Memory violation safety, memory leak safety,...

Some interesting fragments

- Classical fragment

$$\begin{array}{rl} \mathcal{A} ::= & e = e' \mid \mathtt{x} + i \mapsto \{l : e\} \\ & \mid \mathcal{A} \wedge \mathcal{B} \mid \mathcal{A} \rightarrow \mathcal{A} \mid \top \mid \bot \end{array}$$

Some interesting fragments

- Classical fragment

$$\mathcal{A} ::= \quad e = e' \mid \mathtt{x} + i \mapsto \{l : e\} \\ \mid \mathcal{A} \wedge \mathcal{B} \mid \mathcal{A} \rightarrow \mathcal{A} \mid \top \mid \bot$$

- Record fragment

$$\mathcal{A} ::= \quad e = e' \mid \mathtt{x} \mapsto \{l : e\} \\ \mid \mathcal{A} * \mathcal{B} \mid \mathcal{A} \mathrel{-\!\!*} \mathcal{B} \mid \mathtt{emp} \\ \mid \mathcal{A} \wedge \mathcal{B} \mid \mathcal{A} \rightarrow \mathcal{A} \mid \top \mid \bot$$

Some interesting fragments

- Classical fragment

$$\begin{array}{rl} \mathcal{A} ::= & e = e' \mid \ \mathtt{x} + i \mapsto \{l : e\} \\ & \mid \ \mathcal{A} \wedge \mathcal{B} \mid \ \mathcal{A} \rightarrow \mathcal{A} \mid \ \top \mid \bot \end{array}$$

- Record fragment

$$\begin{array}{rl} \mathcal{A} ::= & e = e' \mid \ \mathtt{x} \mapsto \{l : e\} \\ & \mid \mathcal{A} * \mathcal{B} \mid \ \mathcal{A} \mathrel{-\!\!*} \mathcal{B} \mid \ \mathtt{emp} \\ & \mid \ \mathcal{A} \wedge \mathcal{B} \mid \ \mathcal{A} \rightarrow \mathcal{A} \mid \ \top \mid \bot \end{array}$$

- Array fragment

$$\begin{array}{rl} \mathcal{A} ::= & e = e' \mid \ \mathtt{x} + i \mapsto e \\ & \mid \mathcal{A} * \mathcal{B} \mid \ \mathcal{A} \mathrel{-\!\!*} \mathcal{B} \mid \ \mathtt{emp} \\ & \mid \ \mathcal{A} \wedge \mathcal{B} \mid \ \mathcal{A} \rightarrow \mathcal{A} \mid \ \top \mid \bot \end{array}$$

Decidability results

	Classical fragment	Record fragment	Array fragment
SAT	[PSPACE]	[PSPACE]	LTL($\mathbb{N}$)
SAT^{ct}	contains PC^{ct}		
PC	contains Minsky termination [BFN04]		
PC^{ct}	contains reachability without update [IB06]		
MC	contains Minsky termination [BFN04]		
MC^{ct}	reduction to [SAT]	reduction to [SAT]	? ? ?