Model Checking of Parameterized Systems on Weak Memory

David Declerck

Laboratoire de Recherche en Informatique

Université Paris-Sud

October 3rd, 2017

Work supported by French ANR project PARDI (DS0703)
Weak Memory

- order of memory access ≠ interleaving of memory instructions
- we choose a TSO-like model
- reorderings can be prevented using fences
- harder to reason about concurrent programs
Parameterized Systems

- concurrent systems
- unbounded number of processes
- unbounded process-indexed arrays

Example: naive mutual exclusion

```plaintext
type loc = Idle | Want | Crit
array PC[proc] : loc
weak array X[proc] : bool

init (p) { PC[p] = Idle
&& X[p] = False }

unsafe (p1 p2) { PC[p1] = Crit
&& PC[p2] = Crit }

transition t_req ([p])
requires { PC[p] = Idle }
{ PC[p] := Want; X[p] := True }

transition t_enter ([p])
requires { PC[p] = Want && fence(p)
&& forall_other p. X[p] = False }
{ PC[p] := Crit }
```
Our approach

Base framework:
- Model Checking Modulo Theories
- check safety properties of parameterized systems
- assumes a sequentially consistent memory
- relies on a backward reachability algorithm

Our extension:
- add TSO reasoning using an axiomatic model
- maps memory instructions to read/write events
- builds a *global happens-before* relation over events
Backward Reachability Example

\[ PC[#1] = \text{Crit} \land PC[#2] = \text{Crit} \]
Backward Reachability Example
Backward Reachability Example

\[ PC[\#1] = \text{Want} \land PC[\#2] = \text{Want} \]
\[ Rd_x(e_1, \#2, \#1) \land Val(e_1) = \bot \]
\[ Rd_x(e_2, \#1, \#2) \land Val(e_2) = \bot \]
\[ fence(\#2, e_1) \land fence(\#1, e_2) \]

\[ t\_enter(\#1) \]

\[ PC[\#1] = \text{Crit} \land PC[\#2] = \text{Want} \]
\[ Rd_x(e_1, \#2, \#1) \land Val(e_1) = \bot \]
\[ fence(\#2, e_1) \]

\[ t\_enter(\#2) \]

\[ PC[\#1] = \text{Crit} \land PC[\#2] = \text{Crit} \]
Backward Reachability Example

Rd_X(e_1, #2, #1) ∧ Val(e_1) = ⊥
Rd_X(e_2, #1, #2) ∧ Val(e_2) = ⊥
Wr_X(e_3, #2, #2) ∧ Val(e_3) = ⊤
fence(#2, e_1) ∧ fence(#1, e_2)
ghb(e_3, e_1) 
Val(e_2) = Val(e_3)

Rd_X(e_1, #2, #1) ∧ Val(e_1) = ⊥
Rd_X(e_2, #1, #2) ∧ Val(e_2) = ⊥
fence(#2, e_1) ∧ fence(#1, e_2)

Rd_X(e_1, #2, #1) ∧ Val(e_1) = ⊥
fence(#2, e_1)

Backward Reachability Example

PC[#1] = Want ∧ PC[#2] = IDLE
RdX(e_1, #2, #1) ∧ Val(e_1) = ⊥
RdX(e_2, #1, #2) ∧ Val(e_2) = ⊥
WrX(e_3, #2, #2) ∧ Val(e_3) = ⊤

fence(#2, e_1) ∧ fence(#1, e_2)
ghb(e_3, e_1)
Val(e_2) = Val(e_3)
Backward Reachability Example

\[ \text{PC}[\#1] = \text{Want} \land \text{PC}[\#2] = \text{Idle} \]
\[ \text{Rd}_X(e_1, \#2, \#1) \land \text{Val}(e_1) = \bot \]
\[ \text{Rd}_X(e_2, \#1, \#2) \land \text{Val}(e_2) = \bot \]
\[ \text{Wr}_X(e_3, \#2, \#2) \]
\[ \text{fence}(\#2, e_1) \land \text{fence}(\#1, e_2) \]
\[ \text{ghb}(e_3, e_1) \]
\[ t_{\text{req}}(\#2) \]

\[ \text{PC}[\#1] = \text{Want} \land \text{PC}[\#2] = \text{Want} \]
\[ \text{Rd}_X(e_1, \#2, \#1) \land \text{Val}(e_1) = \bot \]
\[ \text{Rd}_X(e_2, \#1, \#2) \land \text{Val}(e_2) = \bot \]
\[ \text{Wr}_X(e_3, \#2, \#2) \land \text{Val}(e_3) = \top \]
\[ \text{fence}(\#2, e_1) \land \text{fence}(\#1, e_2) \]
\[ \text{ghb}(e_3, e_1) \]
\[ t_{\text{enter}}(\#1) \]

\[ \text{PC}[\#1] = \text{Crit} \land \text{PC}[\#2] = \text{Want} \]
\[ \text{Rd}_X(e_1, \#2, \#1) \land \text{Val}(e_1) = \bot \]
\[ \text{fence}(\#2, e_1) \]
\[ t_{\text{enter}}(\#2) \]

\[ \text{PC}[\#1] = \text{Crit} \land \text{PC}[\#2] = \text{Crit} \]