automatic abstraction for complex partial designs

rayna dimitrova, saarland university, germany

partial design verification

- partial design
- distributed system
- communicating components
- interacts with environment

property Ø

implementations for such that the system satisfies Ø

synthesis
derive automatically correct implementation

YES

the implementation can be completed

NO

bug in the implemented components ⇒ revise existing implementation

partial design: bakery mutual exclusion

process a:

10: ticketA := 0;
11: while(true){
12: ticketA := ticketB + 1;
13: await(ticketB = 0 ∨ ticketA < ticketB);
14: critical;
15: ticketA := 0;
}

property:

(whenever pcA = l3, then eventually pcB = l4 and whenever pcA = l4, then eventually pcB = l4)

process b:

m0: ticketB := 0;
m1: while(true){
m2: if ticketB := 0;
m3: ticketB := ticketA;
m4: critical;
m5: ticketB := 0;
}

it is never the case that pcA = l4 ∧ pcB = m4 and

safety property

strengthen to bounded liveness

game model

infinite turn-based game between a component and its environment

tries to violate the property Ø

tries to ensure the property Ø

strategies in the game for the:
- component → implementation
- environment → counterexample

informedness of the component:
- the component player has incomplete information about the global state
- strategy for the component must not depend on information that is not available to it

results

infinite-state concrete game with incomplete information

counterexample-guided abstraction refinement

predicate abstraction w.r.t. finite set of predicates

knowledge-based subset construction

- overapproximate the power of the environment player
- underapproximate the power of the component player
- the abstract component has less information than the concrete

sound abstraction for games under incomplete information

- predicate abstraction
- knowledge-based subset construction

predicate abstraction + knowledge-based subset construction

the predicate pcA=l4 is not observable

sound and complete analysis of abstract counterexamples

- safety properties: abstract strategy for the environment ⇒ strategy tree reduction to satisfiability of a strategy-tree formula

⇒ determine correctly whether an abstract counterexample is concretizable

refinement procedure for games under incomplete information

interpolant computation based on constraint solving

⇒ impose constraints on the interpolants to obtain suitable predicates

⇒ appropriately refine the abstract informedness when this is necessary

ongoing & future work

prototype implementation

- optimize interpolation computation
- extend to other logical theories

application to timed games

- find a suitable symbolic model

distributed partial designs

- make use of component’s locality