The Priced-Timed Maude Tool

Leon Bendiksen and Peter C. Ölveczky

University of Oslo

Priced Systems

Cost

- price, energy use, pollution, ...
- energy use: wireless sensor networks, embedded systems
- planning and scheduling problems
  - task graph scheduling
  - assign landing times and runways to incoming aircraft

Related Work

- Models
  - priced/weighted timed automata
  - priced timed Petri net
- Tool: UPPAAL CORA

Priced-Timed Maude

Priced-Timed Maude extends rewriting logic tool Real-Time Maude to formal modeling, simulation, and model checking of priced (real-time and untimed) systems

Background: Maude

Rewriting logic

- static parts of system specified by equational specification (S, Σ, E)
- dynamic behavior specified by rewrite rules $l : [t]_E \longrightarrow [t']_E$ **if** *cond*

Maude

- rewriting logic tool (simulation; search; LTL model checker)
- natural model of concurrent objects

Background: Objects in Maude

- Class declaration:

  `class` *C* `|` *att1* `:` *s1*`,` `...`, *attn* `:` *sn* `.`

- Object instance is a term

  `<` o `:` *C* `|` *att1* `:` 40`,` `...`, *attn* `:` `"Peter"` `>`

- Distributed state is a multiset of objects and messages

Background: Real-Time Maude

Real-Time Maude

- extends Maude to real-time systems
- successfully applied to systems beyond timed automata
  - wireless sensor network algorithms
  - scheduling algorithms with capacity sharing
  - embedded automotive software
  - IETF and other multicast protocols

Priced-Timed Maude

Priced-Timed Maude

- rules may have cost expressions
  - global cost rules
  - flat object-oriented systems also local cost rules
- analysis:
  - cost-optimal solutions satisfying ...
  - solutions with cost and time constraints
- implemented in Maude as extension of Real-Time Maude

Example: Thermostats

- House with multiple rooms
- . . . each with own thermostat and
- desired temperature interval
- cost of turning on heater depends on temperature

Example: Thermostats (II)

```
(ptomod HEATER is  including STRING .
  including POSRAT-TIME-DOMAIN-WITH-INF + POSRAT-COST-DOMAIN .

  sort OnOff .      ops on off : -> OnOff .      subsort String < Oid .

  class ThStat | state : OnOff, low : PosRat,
                 high : PosRat, temp : PosRat .

  vars T T' : PosRat .  var O : Oid .  var C : Configuration .

  rl [turnOn] :
     < O : ThStat | state : off, temp : T >
    =>
     < O : ThStat | state : on >
    with cost (100 - T) quo 10 + 2 .

  rl [turnOff] :
     < O : ThStat | state : on, high : T, temp : T >
    =>
     < O : ThStat | state : off > .
```

Example: Thermostats (II)

```
  crl [tick] :
      {C} => {delta(C, T)} in time T
          with cost (rate(C) * T) if T <= mte(C) [nonexec] .


  eq delta(< O : ThStat | state : on, temp : T >, T') =
               < O : ThStat | temp : T + (2 * T') > .
  eq delta(< O : ThStat | state : off, temp : T >, T') =
               < O : ThStat | temp : T - T' > .


  eq mte(< O : ThStat | state : on, high : T, temp : T' >) =
              (T - T') / 2 .
  eq mte(< O : ThStat | state : off, low : T, temp : T' >) = T' - T .


  eq rate(< O : ThStat | state : off >) = 0 .
  eq rate(< O : ThStat | state : on >) = 3 .
endptom)
```

Analysis

- Simulating one behavior up to given time and/or cost
- Search for states reachable within given time and cost
- Search for cost-optimal solutions
  - binary search possible
- (Untimed and uncosted) LTL model checking
- Execution w.r.t. selected time sampling strategy for dense time

Example: Simulation

```
Maude> (set tick def 1/2 .)

Maude> (ptfrew
         {< "BedRoom" : ThStat | state : off, low : 62,
                                 high : 70, temp : 66 >
          < "BathRoom" : ThStat | state : off, low : 74,
                                  high : 78, temp : 76 >
          < "Kitchen" : ThStat | state : on, low : 70,
                                 high : 76, temp : 72 >}
        with no time limit with cost <= 30 .)

Result
  ...
```

Example: Optimal Solutions

```
Maude> (find cheapest
          {< "BedRoom" : ThStat | state : off, low : 62,
                                  high : 70, temp : 66 >
           < "BathRoom" : ThStat | state : off, low : 74,
                                   high : 78, temp : 76 >
           < "Kitchen" : ThStat | state : on, low : 70,
                                  high : 76, temp : 72 >}
        =>*
          {< "BathRoom" : ThStat | temp : 78 >  C:Configuration}
        with no time limit .)


Solution
C:Configuration -->
   < "BedRoom" : ThStat | high : 70,low : 62,state : off,temp : 65 >
   < "Kitchen" : ThStat | high : 76,low : 70,state : on,temp : 74 > ;
   REMAINING_ATTRIBUTES_OF_"BathRoom":AttributeSet -->
       high : 78,low : 74,state : on ;
   TIME_ELAPSED:Time --> 1 ;
   TOTAL_COST_INCURRED:Cost --> 10
```

Comparison with Uppaal Cora

- Price-Timed Maude: expressiveness and ease of specification
  - data types
  - avanced OO and communication features
- Uppaal Cora: analytic power
  - priced zones
  - decidable and more efficient
  - finite automaton with clocks and costs
- Multi-priced-timed automata $\subseteq$ Priced-Timed Maude
- . . . not vice versa
  - thermostat example
  - wireless sensor networks
- Priced-Timed Maude significantly slower than Uppaal Cora in Uppaal Cora benchmark

Some Case Studies

- Airplane landing problem: schedule planes onto runways
- Energy task graph scheduling: schedule interdependent tasks onto heterogenous processors
- Subway passenger routing

Concluding Remarks

- Algebraic specifications approach to priced (timed) system
- Systems beyond multi-priced timed automata
- Expressiveness and ease of specification
- Slower analysis than priced timed automata
- Continue to develop tool and analysis methods
  - more mature and efficient
  - non-flat OO systems
  - "cost temporal logic" model checking
- Is the tool really useful?