Abstractions for Analyzing Decision-Based Process Models

B. Lambeau, C. Damas, F. Roucoux and A. van Lamsweerde

MoVES Annual Meeting
December 2009
Modeling critical processes

- Processes & workflows can be safety-critical
  - E.g. healthcare processes
    - Medical errors: 98,000 deaths each year in the US, over 1 million of non-lethal injuries
    - Major cause: complex coordination of clinical tasks

- Critical processes need models for analysis, error anticipation and enactment [Clarke, ICSE’08]

- Building adequate, complete, consistent process models is difficult
Requirements on critical process models

- Close to material provided by stakeholders
  - e.g. medical staff think naturally in terms of...
    - therapy scenarios
    - sequencing of phases composed of tasks
    - decision trees
    - goals & properties on state variables about patient

- Formal semantics, techniques and toolset
  - to support model synthesis, verification and other analyses
Outline

- Guarded High-Level Message Sequence Charts as process modeling language
  - overview of modeling abstractions & semantics

- Abstract decorations for model analysis
  - motivation
  - abstract decoration algorithm

- Analyzing process models
  - verifying task preconditions
  - verifying timing requirements
  - detecting inadequate decisions
Guarded High-Level Message Sequence Charts

- Standard hMSCs + decision nodes
- Task nodes are specified by finer-grained guarded hMSCs or MSC scenarios
- Semantics in terms of Labeled Transition Systems (LTS)
Variables used in decision nodes
1 - Fluents

- Fluents encode task occurrences as propositions

[Giannakopoulou & Magee 2003]

Fluent $F/ = < \text{Init}_{Ffl}, \text{Term}_{Ffl} >$

Fluent $\text{DiagKnown} = < \{\text{Diagnosis}\}, \{\text{FollowUp}\} >$

"DiagKnown holds after Diagnosis and until FollowUp"
Variables used in decision nodes

2 - Tracking variables

- Variables tracking environment quantities
  - not directly observable by process agents

\[ \text{TrackingVar } TV = \{ \text{Update}_{TV} \} \]

- Value not deterministically known (unlike fluents)
  - after BloodTest the platelet level may be low or not

```
BloodTest

PlateletLow? - no → Wait 5 days

- yes → Chemotherapy

TrackingVar PlateletLow = \{ \text{BloodTest} \}

"Platelet level is determined during BloodTest"
```
Other abstractions

- Duration, doses, costs attached to tasks
  - Radiotherapy Session
    - dose: 5 Gy
    - duration: 15 minutes

- Task preconditions
  - Radiotherapy Session
    - for safety goal *Avoid [IrradiatedTwice]*
      - precondition: $\neg$ Irradiated
What kind of model analysis?

- Checking task preconditions
  - "Radiotherapy may not take place if patient has already been irradiated in the past"

- Checking decision adequacy
  - "Platelet level in patient record should reflect the real patient's platelet level each time a related decision is taken"

- Checking non-functional requirements about timing, dosage, cost, etc
  - "Complete treatment may not exceed 40 days"
  - "Radiotherapy treatment must deliver exactly 45 Gy"
From process models to analyzable LTS

- **Guarded hMSCs**
  - process modeling language

- **Guarded LTS** =
  - Labeled Transition System with guards or events on transitions

- **LTS**
  - trace semantics
  - trace-based model checking

Abstract decorations for model analysis

- Computing the set of LTS traces accepted by a guarded hMSC ...
  - enables verification of temporal logic properties ...
  - ... but is time consuming ...
  - no support for reasoning about doses, timing, etc.

- Useful analyses can be performed on guarded LTS
  - avoid explicit trace enumeration
  - symbolically decorate states using well-chosen abstractions for specific analyses
From guarded hMSC to guarded LTS

- Task\textsubscript{start} and Task\textsubscript{end} events are introduced to support more accurate fluent definitions
  - refinements yield events between them
- "Bricks" are connected with $\tau$ transitions
  - mapping preserved between g-hMSC tasks and g-LTS states for analysis feedback
Decorating g-LTS models: a generic decoration algorithm

- Concrete decorations to be defined as instantiations over bounded lattices
  - initial decoration for initial state

- Abstract interpretation algorithm
  - generic propagation rule computes the effect of a transition on the decoration of a source state
  - generic decorations accumulated through supremum operator

- When fixpoint is reached ...
  - state decoration = accumulation of decorations contributed by all paths leading to the state
Process model analysis
1- Verifying task preconditions

- Decorations are Boolean expressions on the set of fluents and tracking variables
- Instantiated propagation rule handles guards & events

```
Propagate(deco, edge):
    if label(edge) is a guard
        return deco ∧ label(edge)
    else /* ... is an event */
        return deco | label(edge)
```

- Lattice supremum operator = disjunction on Bool lattice
- Check ...

  for each source state of a Task\(_{\text{start}}\) event:
  decoration (source) \(\models\) Task\(_{\text{PRE}}\) ?
Process model analysis
2- Verifying temporal requirements

- Decorations are functions
  
  \[ \text{timeWindow} : 2\Phi \rightarrow \mathbb{P}[\mathbb{R}^+] \]
  
  - \( \Phi \): set of all fluents & tracking variables
  - codomain: union of time intervals
  - map specific process instances to time points where the state can be visited

- Propagation rule
  - on guards: function restriction
  - otherwise interval is shifted to account for task duration

- Supremum
  - union of time intervals
Verifying temporal constraints: example

- Max 40 days -- ok!
- With similar technique: irradiation dose always exactly 45 Gy
3- Detecting inadequate decisions due to inaccurate information

- **Accuracy fluents** associated with tracking variables

  fluent `PlateletLowAccurate` =
  
  `< {BloodTest}, {ChemoTherapy} >` initially false

  "The platelet level in the patient's record is accurate after BloodTest but should be considered inaccurate after ChemoTherapy"

- **Analysis:** generate state invariants as $g$-LTS decorations
  
  - `PlateletLowAccurate` must always hold in source states of any guarded transition involving `PlateletLow`
4- Detecting inadequate decisions due to outdated information

- Timed accuracy fluents associated with tracking variables

  fluent PlateletLowAccurate =
  \(<\{\text{BloodTest}\}, \{\text{ChemoTherapy}\}\)> initially false
  duration 7 days

  "The platelet level in the patient record is also considered outdated 7 days after a BloodTest"

- Analysis: decorate g-LTS with state invariants while propagating timing information
  - blend of previous decoration types
  - same kind of check
Ongoing work

- In case of property violation, no counterexample generated so far
  - often required to understand the problem and correct the model

- Algorithm instantiations for other kind of analyses
  - e.g. resource usage

- Domain properties to be introduced in the model for simplifying derived information such as preconditions

- Goals underlying tasks should be made explicit