Formalizing Non-Concurrent UML State Machines Using Colored Petri Nets

Étienne André, Christine Choppy, Kaïs Klaï

Laboratoire d’Informatique de Paris Nord
Université Paris 13, Sorbonne Paris Cité, France
Introduction

Context: Verification of Complex Systems

- Need for early bug detection
  - Bugs discovered when final testing: expensive
  - Need for a thorough modeling phase
UML Behavioral State Machines

- Transition systems used to express the behavior of dynamic systems
- Specified in [OMG, 2009] (version 2.2)
- Widely used in the industry
- Semantics not formally expressed
  - Informal specification in [OMG, 2009]
  - Not directly suitable for formal methods
Example of a CD Player [Zhang and Liu, 2010]

- **Features**
  - A *hierarchy* of simple and composite states
  - Transitions (including inter-level) with *events*
  - Entry (*find track start*) and do (*play track*) *behaviors*
  - Global *variables* (*present* and *track*)
  - History pseudostate (*H*)
Example of a CD Player (cont.)

- This example is simple
  - Few states, few events, few variables
  - No concurrency
  - No exit behavior

And still... Can we ensure the following?
- When in PLAYING, there is a CD in the player
- When in PLAYING, the track number is always between 1 and $trackCount$

Not easy to guarantee! (So what about larger case studies...?)

Étienne ANDRÉ (Paris 13)
Formalizing UML State Machines
27th August 2012
Example of a CD Player (cont.)

- This example is simple
  - Few states, few events, few variables
  - No concurrency
  - No exit behavior

- And still... Can we ensure the following?
  - “When in PLAYING, there is a CD in the player”
  - “When in PLAYING, the track number is always between 1 and trackCount”
Example of a CD Player (cont.)

- This example is simple
  - Few states, few events, few variables
  - No concurrency
  - No exit behavior

- And still... Can we ensure the following?
  - “When in PLAYING, there is a CD in the player”
  - “When in PLAYING, the track number is always between 1 and trackCount”

- Not easy to guarantee!
  (So what about larger case studies...)
Main Goal

Goal

“Give a semantics to UML state machines so as to allow for formal verification.”
Main Goal

Goal

“Give a semantics to UML state machines so as to allow for formal verification.”

- We choose here to automatically translate UML state machines to colored Petri nets (CPNs)

- Set of considered constructs
  - Hierarchy of composite states
  - Inter-level transitions
  - Entry, do, exit behaviors with global variables
  - History pseudostates
  - No concurrency (no fork, join, synchronization)
Related Works

- Semantics directly defined
  - [Jin et al., 2004, Dubrovin and Junntila, 2007]

- Translations of ULM state machines to CPNs
  - [Pettit IV and Gomaa, 2006, Lian et al., 2008, Choppy et al., 2011]

- Translations of ULM state machines to other formalisms
  - SPIN [Latella et al., 1999], SMV [Clarke and Heinle, 2000], CSP# [Zhang and Liu, 2010], etc.
**Related Works**

- **Semantics directly defined**
  - [Jin et al., 2004, Dubrovin and Juntila, 2007]

- **Translations of ULM state machines to CPNs**
  - [Pettit IV and Gomaa, 2006, Lian et al., 2008, Choppy et al., 2011]

- **Translations of ULM state machines to other formalisms**
  - SPIN [Latella et al., 1999], SMV [Clarke and Heinle, 2000], CSP# [Zhang and Liu, 2010], etc.

- **Limitations**
  - Almost always *restrictive subset* of syntactic constructs
  - Often *limited specification* of properties
Outline

1. Colored Petri Nets
2. Translation
3. Application to the CD Player
4. Perspectives
Outline

1. Colored Petri Nets
2. Translation
3. Application to the CD Player
4. Perspectives
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive graphical representation
  - A formal semantics
  - Powerful tools to test and check the model

Example: ADVD renting machine

- Customer's coins
- Earned coins
- D VDs available
- D VDs on loan
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive graphical representation
  - A formal semantics
  - Powerful tools to test and check the model

- Example: A DVD renting machine

![Diagram of a DVD renting machine](image)
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive **graphical representation**
  - A formal semantics
  - Powerful tools to test and check the model

- Example: A DVD renting machine

```
Customer’s coins

Earned coins <-> DVDs available

DVDs on loan
```
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive graphical representation
  - A formal semantics
  - Powerful tools to test and check the model

- Example: A DVD renting machine

Customer’s coins

Earned coins → DVDs available

DVDs on loan
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive graphical representation
  - A formal semantics
  - Powerful tools to test and check the model

- Example: A DVD renting machine

![Diagram of a DVD renting machine](attachment:diagram.png)
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive graphical representation
  - A formal semantics
  - Powerful tools to test and check the model

- Example: A DVD renting machine
  
  ![Diagram of a DVD renting machine]

  - Customer’s coins
  - Earned coins
  - DVDs available
  - DVDs on loan
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive **graphical representation**
  - A **formal semantics**
  - **Powerful tools** to test and check the model

- Example: A DVD renting machine

![Colored Petri Net Diagram](image-url)

- Customer’s coins
- Earned coins
- DVDs available
- DVDs on loan
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive graphical representation
  - A formal semantics
  - Powerful tools to test and check the model

- Example: A DVD renting machine

```
Customer’s coins

Earned coins  DVDs available

DVDs on loan
```
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive **graphical representation**
  - A **formal semantics**
  - **Powerful tools** to test and check the model

- Example: A DVD renting machine

```
Customer’s coins

Earned coins  DVDs available

DVDs on loan
```
Petri Nets [Petri, 1962]

- Advantages of Petri nets
  - Detailed view of the process with an expressive **graphical** representation
  - A **formal semantics**
  - **Powerful tools** to test and check the model

- Example: A DVD renting machine

Customer’s coins

Earned coins

<table>
<thead>
<tr>
<th>DVDs available</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVDs on loan</td>
</tr>
</tbody>
</table>
Colored Petri Nets [Jensen and Kristensen, 2009]

- Extension of Petri nets with **colors**
  - Tokens and places have a **type** (“color set”)
  - Arcs are labeled with **expressions**
  - Transitions can have a **guard**
Colored Petri Nets [Jensen and Kristensen, 2009]

- Extension of Petri nets with colors
  - Tokens and places have a type (“color set”)
  - Arcs are labeled with expressions
  - Transitions can have a guard

- Example: A more complex version of the DVD renting machine

![Diagram of a colored Petri net model of a DVD renting machine with transitions and additional labels for customers and money earned.]

Legend
- Customers
- Money earned
- DVDs available
- DVDs on loan
Colored Petri Nets [Jensen and Kristensen, 2009]

- Extension of Petri nets with colors
  - Tokens and places have a type ("color set")
  - Arcs are labeled with expressions
  - Transitions can have a guard

- Example: A more complex version of the DVD renting machine

Legend
- Customers
- Money earned
- DVDs available
- DVDs on loan
Colored Petri Nets [Jensen and Kristensen, 2009]

- Extension of Petri nets with colors
  - Tokens and places have a type ("color set")
  - Arcs are labeled with expressions
  - Transitions can have a guard

- Example: A more complex version of the DVD renting machine
Colored Petri Nets [Jensen and Kristensen, 2009]

- Extension of Petri nets with colors
  - Tokens and places have a type (“color set”)
  - Arcs are labeled with expressions
  - Transitions can have a guard

- Example: A more complex version of the DVD renting machine

```
NAMExINT 1'(Alice, 24€) ++ 1'(Bob, 20€)
INT [m ≥ p] e + p c, m - p
FILMxINT 2'(Satan Tango, 12€) ++ 1'(Un retour, 10€)
FILMxINT 1'(Rashōmon, 6€)
```

Legend
- Customers
- Money earned
- DVDs available
- DVDs on loan
Colored Petri Nets [Jensen and Kristensen, 2009]

- Extension of Petri nets with colors
  - Tokens and places have a type ("color set")
  - Arcs are labeled with expressions
  - Transitions can have a guard

- Example: A more complex version of the DVD renting machine

---

Diagram:

```
Colored Petri Nets

1'(Alice, 24€) + 1'(Bob, 8€)
INT 1'(18€)

FILMxINT 1'(Satan Tango, 12€) + 1'(Un retour, 10€)

Legend

- Customers
- Money earned
- DVDs available
- DVDs on loan
```

Étienne ANDRÉ (Paris 13)  Formalizing UML State Machines  27th August 2012
Colored Petri Nets [Jensen and Kristensen, 2009]

- Extension of Petri nets with colors
  - Tokens and places have a type ("color set")
  - Arcs are labeled with expressions
  - Transitions can have a guard

- Example: A more complex version of the DVD renting machine

![Colored Petri Net Diagram]

Legend

- Customers
- Money earned
- DVDs available
- DVDs on loan

Étienne ANDRÉ (Paris 13)  Formalizing UML State Machines  27th August 2012  11 / 22
Colored Petri Nets [Jensen and Kristensen, 2009]

- Extension of Petri nets with colors
  - Tokens and places have a type ("color set")
  - Arcs are labeled with expressions
  - Transitions can have a guard

- Example: A more complex version of the DVD renting machine

```
1'(Alice, 24€) ++ 1'(Bob, 8€)
```

```
INT 1'(18€)
```

```
FILMxINT 2'(Satan Tango, 12€)
```

```
++ 1'(Un retour, 10€)
```

```
1'(Rashōmon, 6€)
```

Legend
- Customers
- Money earned
- DVDs available
- DVDs on loan
Outline

1. Colored Petri Nets
2. Translation
3. Application to the CD Player
4. Perspectives
Translating States and Behaviors

- Each simple state, final state, and history pseudostate is translated into a CPN place
  - Composite states not translated, only their simple substates are

- Each event and behavior is translated into a CPN transition

- Additional CPN places, transitions and arcs needed
  - Link between behaviors and states, etc.
  - Transitions
Typing

- Only one (colored) token in the resulting CPN
  - Possible because of non-concurrency

- This token carries:
  - The value of the global variables
  - The name of the latest substate for states containing a history pseudostate

- Example: CD Player
  - 2 global variables (present and track), one history pseudostate
  - Type of the token: \{true, false\} × N × \{PLAYING, PAUSED\}
  - Example of value: (true, 2, PAUSED)
Translating Transitions

- A possible translation
Translating Transitions

- A possible translation

- Main problem: factoring
  - For each behavior in UML, we want only one CPN transition
  - Idea: use synchronization places
Translating Global Variables

- Guards involving variables

```plaintext
 PLAYING
   play [present = true]

 NONPLAYING
```

---

**Étienne ANDRÉ (Paris 13)**

Formalizing UML State Machines

27th August 2012 16 / 22
Translating Global Variables

- Guards involving variables
  - PLAYING
    - play [present = true]
  - NONPLAYING

- Transitions updating variables
  - CLOSED
    - load / present := true; track++
  - OPEN
Translating History States

- For each transition leading to a substate, update the token

- For each history pseudostate, add CPN transitions to places corresponding to all substates, with the adequate guard

- Note: this mechanism combines with the mechanism for behaviors (in case entry behaviors should be performed)
Translating History States

- For each transition leading to a substate, update the token

- For each history pseudostate, add CPN transitions to places corresponding to all substates, with the adequate guard

- Note: this mechanism combines with the mechanism for behaviors (in case entry behaviors should be performed)
Outline

1. Colored Petri Nets
2. Translation
3. Application to the CD Player
4. Perspectives
where vh stands for p, t, h

Partial scheme

(See full scheme in [André et al., 2012])
Application to the CD Player (2/2)

😊 Scheme graphically more complex than the original diagram
  - But still possible to read visually

😊 But automated verification now possible
Application to the CD Player (2/2)

Scheme graphically more complex than the original diagram
- But still possible to read visually

But automated verification now possible

“When in PLAYING, there is a CD in the player”
Application to the CD Player (2/2)

😊 Scheme graphically more complex than the original diagram
  ✔ But still possible to read visually

😊 But automated verification now possible
  ✔ “When in PLAYING, there is a CD in the player”
Scheme graphically more complex than the original diagram
- But still possible to read visually

- But automated verification now possible

 √ “When in PLAYING, there is a CD in the player”
 √ “When in PLAYING, the track number is always between 1 and trackCount”
Application to the CD Player (2/2)

😊 Scheme graphically more complex than the original diagram
  - But still possible to read visually

😊 But automated verification now possible
  ✓ “When in PLAYING, there is a CD in the player”
  ✓ “When in PLAYING, the track number is always between 1 and trackCount”
Application to the CD Player (2/2)

😊 Scheme graphically more complex than the original diagram
   - But still possible to read visually

😊 But automated verification now possible
   - “When in PLAYING, there is a CD in the player”
   - “When in PLAYING, the track number is always between 1 and trackCount”

(Properties verified using CPN Tools)
Outline

1. Colored Petri Nets
2. Translation
3. Application to the CD Player
4. Perspectives
Future Work

- Implementation and evaluation

- Extension to concurrent UML state machines
  - Use “global places” to encode global variables and history states
  - Use the mechanisms from [Choppy et al., 2011] to encode fork / join constructs

- Extension to full UML state machines 2.3 [OMG, 2010]
  - Including time issues

- Proof of equivalence with a semantics
  - Need to give a formal semantics to UML state machines first
References I


A method for describing the syntax and semantics of UML statecharts.

Automatic verification of a behavioural subset of UML statechart diagrams using the
SPIN model-checker.

Simulation-based analysis of UML statechart diagrams: methods and case studies.
*Software Quality Journal*, 16(1):45–78.

OMG (2009).
http://www.omg.org/spec/UML/2.2/Superstructure/PDF/.

OMG (2010).
