Formal verification of UML state diagrams: a Petri net based approach

Christine Choppy, Kais Klai, Hacene Zidani
LIPN, Universite Paris 13, France

UML& FM 2010

Outline

- Introduction
- Approach
- Related works
- Example
- Transformation Algorithm
- Formal verification
- Conclusion & Perspectives

Introduction

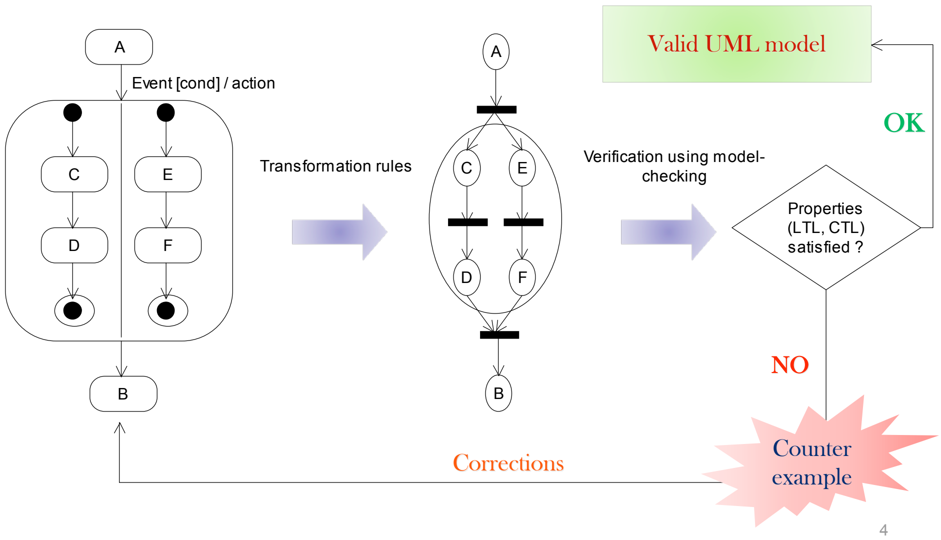
Goal

- Combine expressive power and formal verification for complex systems design
- UML, expressive modelling language:
  - graphical
  - semi formal
  - widespread in industry
- UML semantics lacks formality

- Associated formal models are proposed (here state diagrams are studied)
- So as to use mathematical reasoning & tools

- Here: Hierarchical Coloured Petri Nets (HCPNs) and model checking tools
- Modularity: how to link HCPNs associated with the different state diagrams

Approach



Related works

Limits

- Shatz et al.:
  - Size of the associated coloured Petri net
  - Complex format of arc labels (<type, flag>)
  - The way a composite state is transformed
  - Add one place for each event

Large size of the generated Petri net

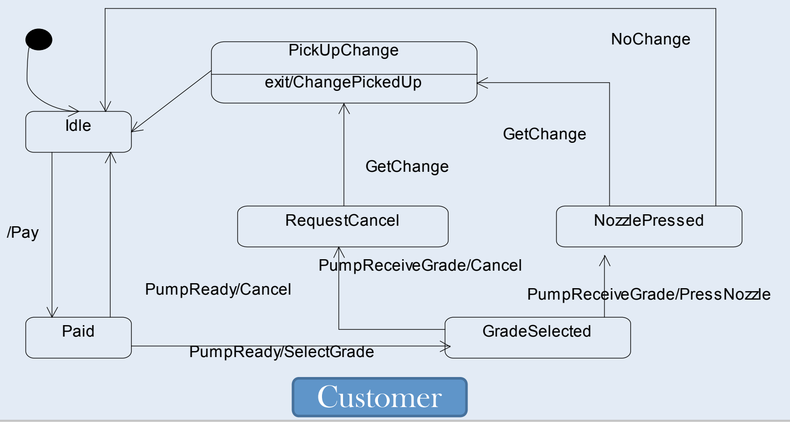
- Pettit & Gomaa :

Integration of CPNs in OO architecture

- Here:
  - Subset of state diagrams: actions generate events, ...
  - Modular transformation, hierarchical coloured Petri nets
  - Properties expressed in temporal logic (LTL, CTL), model checking

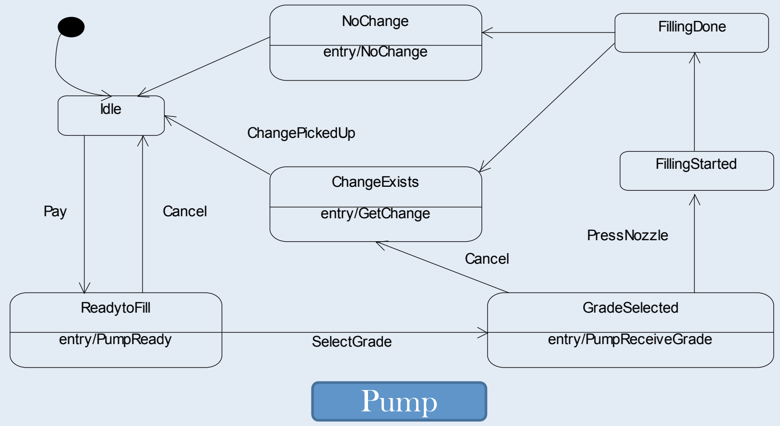
Example

Gas station (Customer state diagram)



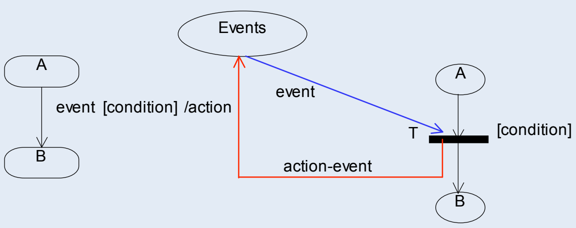
Example

Gas station (Pump state diagram)



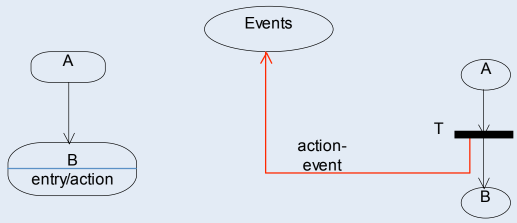
Algorithm

Main transformation (general)



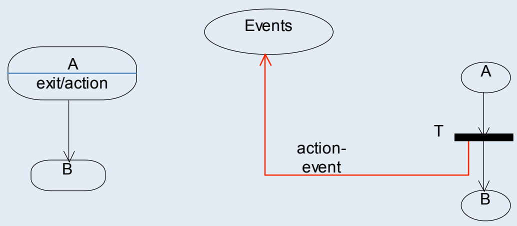
Algorithm

Main transformation (entry action)



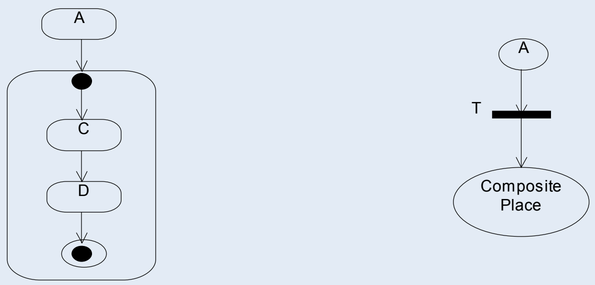
Algorithm

Main transformation (exit action)



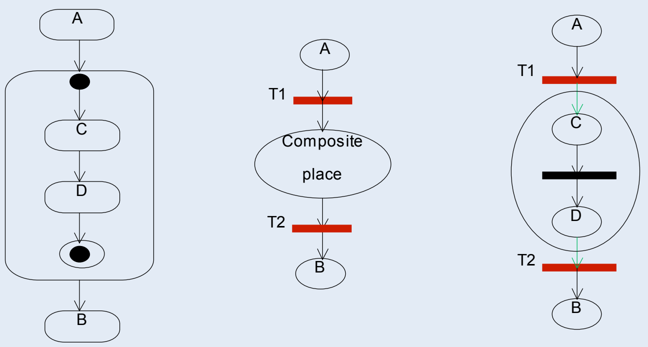
Algorithm

Main transformation (composite state)



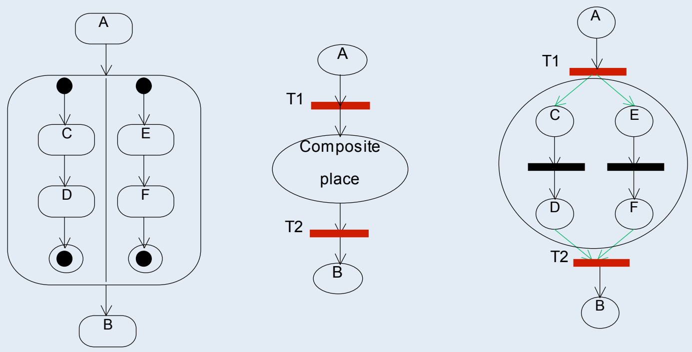
Algorithm

Composite state transformation (sequential)



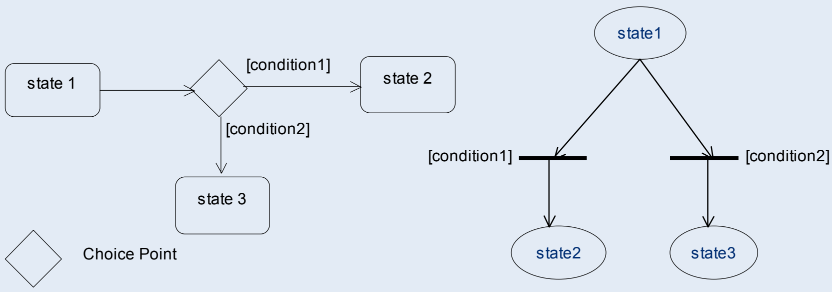
Algorithm

Composite state transformation (concurrent)



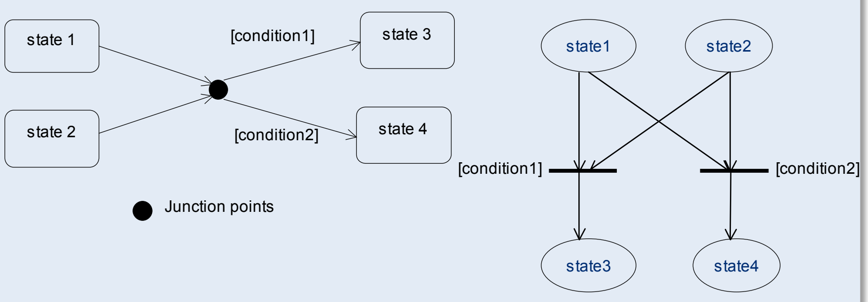
Algorithm

Transformation of a choice point

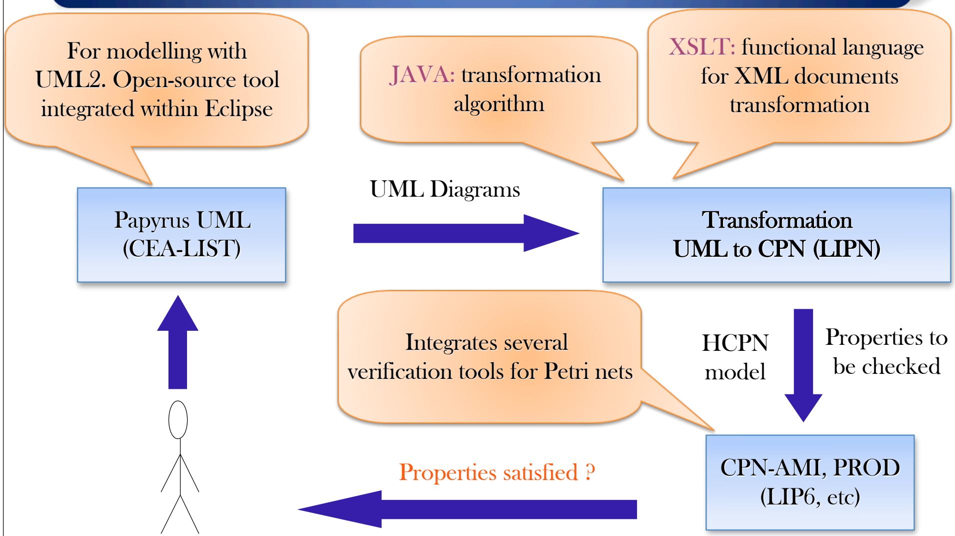


Algorithm

Transformation of a junction point

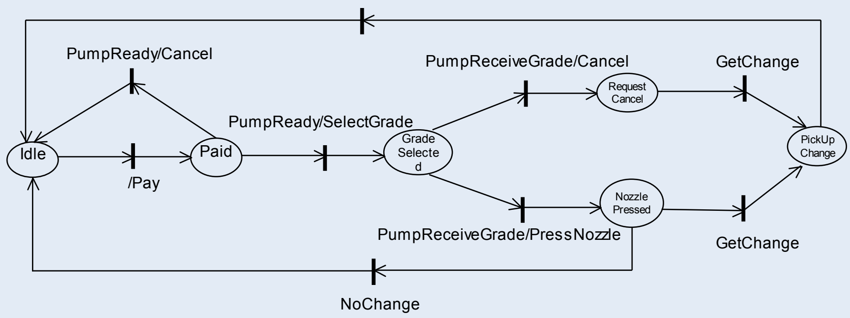


Tool implementation



Example

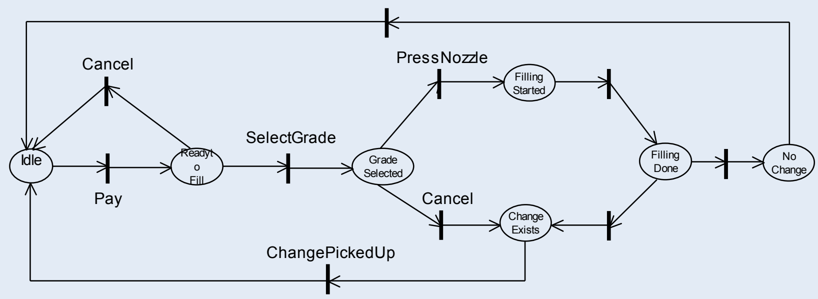
Gas station (Customer Petri net 1)



Customer

Example

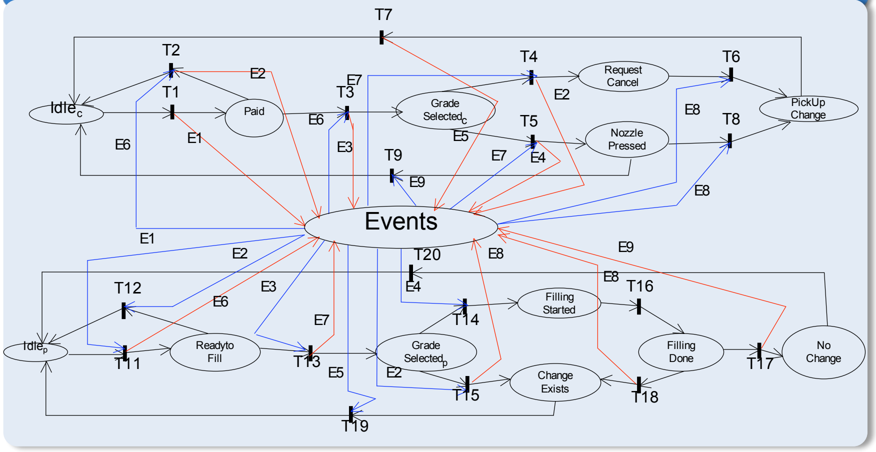
Gas station (Pump Petri net 2)



Pump

Example

Gas station (Petri net 3)



Formal Verification

Gas station example (1)

The customer may choose the gas grade
after (s)he paid

$\neg$ GradeSelected$_c$ **U** Paid

Property satisfied

Formal Verification

Gas station example (2)

If the customer paid (s)he eventually will be prompted
to choose the gas grade

G (Paid ⇒ **F** GradeSelected$_c$)

Property not satisfied !

Formal Verification

Gas station example (3)

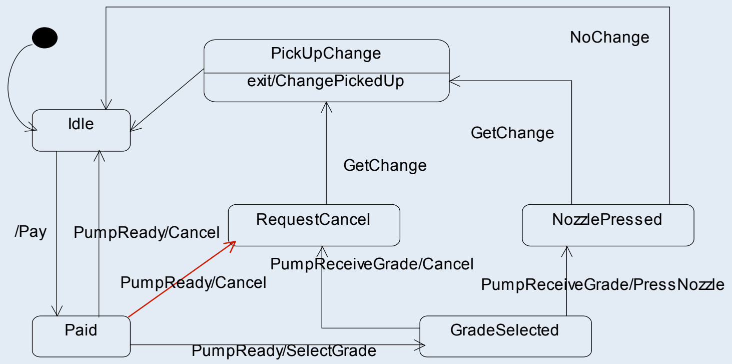
The customer may always cancel the request

AG (**EF** RequestCancel)

Property satisfied

UML model update

State diagram update (customer)



Customer

Update of the properties

Correction of the not satisfied properties

If the customer paid (s)he will eventually be prompted
to choose the gas grade if (s)he did not cancel the request

G [Paid ⇒ **F** (GradeSelected$_c$ ∨ RequestCancel)]

Property satisfied

Conclusion et Perspectives

Conclusion

- Framework for the formal verification of UML state diagrams
- New transformation algorithm of UML state diagrams to hierarchical coloured Petri nets
- Modular transformation
- Restricted to a subset of state diagrams (event generating actions, ...)
- Size of the net
- Formal verification to check properties of the state diagrams
- Gas station example: some expected properties not satisfied led to correct the model
- Real size case study (part of air traffic control system) underway

Conclusion et Perspectives

Perspectives

- Other case studies
- Extend transformation algorithm to process other concepts of state diagrams (other actions, history, etc)
- Take other languages for expressing properties into account (Object Constraint Logic): translate OCL formula to formula used in the model checking tool, and translate back counter-examples when property not satisfied
- Formal verification of other UML diagrams (collaboration, activity, ...etc)