

# On Conditions for Modular Verification in Systems of Synchronising Components

*Peter Drábik*, Andrea Maggiolo-Schettini, Paolo Milazzo

Dipartimento di Informatica, Università di Pisa, Italy

Pułtusk – CS&P 2011 – September 28, 2011

# Formal verification

- Formal verification – techniques for formally analysing properties of a system
- Model checking – useful and established method
- Exploring all possible behaviours – in many cases infeasible due to complexity
- State explosion problem
- A method to avoid it – decomposition of the system and application of modular verification

# Modular verification

- Inferring global properties from properties of components
- Often properties concern only a small portion
- Isolate a minimal fragment necessary for verification
- Property preservation, from subprograms to complete programs
- Class of properties preserved are ACTL – the universal fragment of CTL

# Outline

- Formalisms with multi-way synchronisation (for modelling purposes)
- Synchronisation is seen as a feature of the semantics
- General type of synchronisation – modular verification cannot be applied
- Specific type of synchronisation is required – conditions identified
- Complete synchronisation – the semantics of the formalism must permit the simultaneous execution only of component moves that reference each other in the synchronisation requirements
- Application to other formalisms

## Property preservation on LTSs (1)

- Set  $I$  of component indices, atomic propositions  $AP_i$  for  $i \in I$
- $I$ -states over APs, tuples  $[a_1, \dots, a_n]$
- transitions between  $I$ -states labelled by indices
- Path, fullpath, fair path
- Path projection onto  $J \subseteq I$ : project states and labels

### Example

$$[A, B, C] \xrightarrow{3} [A, B, \neg C]$$

## Property preservation on LTSs (2)

- Simulation relation between LTSs  
Relation of whole system and an approximation of its part
- $I$ -structure  $M_1$  and  $J$ -structure  $M_2$  ( $J \subseteq I$ )
- $M_2$  simulates  $M_1$  iff for each path  $\pi$  in  $M_1$ ,  $M_2$  contains  $\pi \upharpoonright J$
- ACTL<sup>-</sup>:  $p \in AP$ ,  $\neg p$ ,  $f \wedge g$ , ...,  $AG f$ ,  $A[f U g]$ ,  $A[f U_w g]$

### Simulation implies property preservation

If  $M_2$  simulates  $M_1$  then whenever  $M_2 \models f$  then  $M_1 \models f$

# Modular verification of programs

- Simulation can be used for modular verification of ACTL<sup>-</sup> properties
- For programs with  $I$ -structures as semantics
- Program  $P_1$  with semantics  $M_1$ ,  $f$  property from ACTL <sub>$J$</sub>
- Instead of constructing  $M_1$  and verifying  $f$  we construct a  $J$ -structure  $M_2$  such that
  - 1  $M_2$  simulates  $M_1$ , which implies the preservation of ACTL <sub>$J$</sub>  formulae
  - 2  $M_2$  is constructed directly, avoids state space explosion
- We construct  $M_2$  directly by applying a syntactic projection to program  $P_1$  and then construct its semantics

# Modular verification of programs without synchronisation

- Program is a collection of automata (indexed by  $I$ )
- Automata moves contain conditions on states of other automata

## Example

$$\neg C\_bar \xrightarrow{A\_bar, \neg B\_bar} C\_bar$$

- Semantics – an  $I$ -structure
- Syntactic projection onto  $J$  – leaves out components that are not in  $J$  and removes references in the conditions to them

## Result

Syntactic projection guarantees simulation between the respective semantics

## ... with synchronisation

- Synchronisation
  - A characteristic feature of many kinds of systems
  - Necessary as a modelling primitive – automaton expresses for each move conditions on moves of other automata, without which it cannot be executed
  - Taken into account by the formal semantics
  - Impact of the choices on modular verification
- Extension of the previous formalism by synchronisation
- Analogous simple syntactic projection
- Prove the simulation in the style of the previous formalism
- This is not possible in general, we investigate the conditions on semantics that allow to obtain the result

## Synchronisation conditions

- Formalism – change synchronisation conditions
- State based conditions
- Semantics – transitions involve execution of synchronised moves, all the others are idle

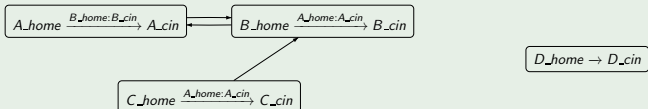
### Example

$$\begin{array}{l}
 A\_home \xrightarrow{B\_home:B\_cin} A\_cin, \quad B\_home \xrightarrow{A\_home:A\_cin} B\_cin, \\
 C\_home \xrightarrow{A\_home:A\_cin} C\_cin, \quad D\_home \rightarrow D\_cin
 \end{array}$$

# Synchronisation

- A synchronisation is a set of moves that are executed simultaneously
- Intuitively three requirements:
  - 1 moves are from distinct automata
  - 2 the requirements of each move are satisfied
  - 3 only related moves (atomicity)
- the third condition formalisation
  - directed graph where vertices are moves, edges references
  - weak connectivity

## Example



## Synchronisation and simulation

- syntactic projection analogous to before
- observation: the simulation result doesn't hold

### Example

$$\begin{array}{l}
 A\_home \xrightarrow{B\_home:B\_cin} A\_cin, \quad B\_home \xrightarrow{A\_home:A\_cin} B\_cin, \\
 C\_home \xrightarrow{A\_home:A\_cin} C\_cin, \quad D\_home \rightarrow D\_cin
 \end{array}$$

- with the current definition of semantics, simulation doesn't hold
- since the projection is fixed, we can study additional conditions on the semantics to guarantee simulation
- change the definition of synchronisation

## Synchronisation and simulation

- syntactic projection analogous to before
- observation: the simulation result doesn't hold

### Example

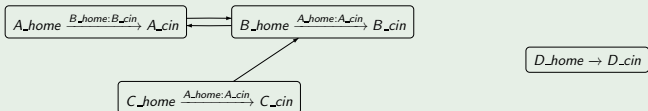
$$\begin{array}{l}
 B_{home} \longrightarrow B_{cin}, \\
 C_{home} \longrightarrow C_{cin}, \quad D_{home} \rightarrow D_{cin}
 \end{array}$$

- with the current definition of semantics, simulation doesn't hold
- since the projection is fixed, we can study additional conditions on the semantics to guarantee simulation
- change the definition of synchronisation

## Redefining synchronisation

- what we want is: if  $MOV$  is a synchronisation of moves from  $P_I$  then  $MOV \upharpoonright J$  is a synchronisation of moves from  $P_J = P_I \upharpoonright J$
- Requirement on synchronisation:
  - ① moves are from distinct automata
  - ② the requirements of each move are satisfied
  - ③ only related moves (weakly connected reference graph)
- Violation clearly stems from the third point
- Necessary to impose a stronger condition on the graph

### Example



## Stronger graph condition

- Let's call  $Prop$  the property of graph  $G_{MOV}$
- Two formal requirements
  - 1 if  $Prop(G_{MOV})$  then  $G_{MOV}$  is weakly directed
  - 2 if  $Prop(G_{MOV})$  then  $Prop(G_{MOV \downarrow J})$
- Projection of a synchronisation removes nodes and their induced edges producing an induced subgraph
- $Prop$  is induced hereditary: if  $Prop$  holds for  $G$  then it holds for all induced subgraphs  $H$  of  $G$
- induced hereditary: planarity, outerplanarity, bipartity, acyclicity, having max degree, completeness
- not ind hereditary: weak connectivity, connectivity, strong connectivity

# Complete synchronisation

- We prove that  $Prop(G) \leftrightarrow Complete(G)$
- Synchronisation has to consider only moves that reference each other.
- We call such a synchronisation complete
  - ① moves are from distinct automata
  - ② the requirements of each move are satisfied
  - ③ only related moves (**complete** reference graph)
- Implications:
  - Complete synchronisation is preserved by projection
  - Simulation is proved for semantics of sync-programs with complete synchronisation
  - Modular verification of  $ACTL_J$  formulae can be performed

# Summary

- Property preservation as an approach to modular verification
- Reduction of property verification time for formal models
- Formalisms with multi-way synchronisation
- Automata-based language, each move explicitly indicates all moves of other automata without which it is not willing to be performed
- Intuition – minimalist synchronisation
- Modular verification requires a specific type of synchronisation
- Identified the necessary condition
- Complete synchronisation
- To avoid synchronisation of two components through a third party that could be removed by projection operation

## Application & Discussion

- Generalise the principles to other formalisms
- Semantics has the complete synchronisation schema
  - 1 moves from distinct automata
  - 2 for each move its condition is satisfied
  - 3 each move is requesting synchronisation with all the participating moves
- CSP – even stronger condition – execute whenever an action is in the alphabet
- CCS – complementary channels, always in pairs, not applicable
- Reduced program is obtained by a simple and intuitive projection operator
- To apply to other types of synchronisation, one must consider more sophisticated forms of syntactic projections

## PhD in Pisa

We would like to announce **8 grants** (+3 pending) at the PhD programme in Computer Science of the School Galileo Galilei.

The **deadline** for applications is: **12th October 2011** The interview will take place on: 24th October 2011

For further details, visit:

<http://phd.di.unipi.it/wp/Bando2011.aspx>

or start your visit from <http://dottorato.unipi.it/>