

# TAPAs: a Tool for the Analysis of Process Algebras

Francesco Calzolari Rocco De Nicola Michele Loreti  
Francesco Tiezzi

Università degli Studi di Firenze  
Dipartimento di Sistemi e Informatica

2<sup>nd</sup> Workshop on Teaching Concurrency  
25 June 2007 — Siedlce, Poland

# Process Algebras

Process Algebras are a set of mathematically rigorous languages

- provide a number of constructs for system descriptions
- are equipped with an operational semantics that describes systems evolution

Often come with observational mechanisms that permit identifying systems that cannot be taken apart by external observations

- In some cases they come also equipped with sets of axioms, that capture the relevant identifications.

There has been huge amount of research work on process algebras carried out in the last 25 years:

- CCS, CSP, ACP, ...

# Process Algebras

Process Algebras are a set of mathematically rigorous languages

- provide a number of constructs for system descriptions
- are equipped with an operational semantics that describes systems evolution

Often come with observational mechanisms that permit identifying systems that cannot be taken apart by external observations

- In some cases they come also equipped with sets of axioms, that capture the relevant identifications.

There has been huge amount of research work on process algebras carried out in the last 25 years:

- CCS, CSP, ACP, ...

# Process Algebras

Process Algebras are a set of mathematically rigorous languages

- provide a number of constructs for system descriptions
- are equipped with an operational semantics that describes systems evolution

Often come with observational mechanisms that permit identifying systems that cannot be taken apart by external observations

- In some cases they came also equipped with sets of axioms, that capture the relevant identifications.

There has been huge amount of research work on process algebras carried out in the last 25 years:

- CCS, CSP, ACP, ...

# Process Algebras: System Verifications

Verification of concurrent system within the process algebraic approach is performed:

- by resorting to behavioural equivalences
- by checking that processes enjoy properties described by temporal logics formulae

# Process Algebras: System Verifications

Verification of concurrent system within the process algebraic approach is performed:

- by resorting to behavioural equivalences
- by checking that processes enjoy properties described by temporal logics formulae

## Behavioral Equivalences

- Two descriptions of a given system are considered:
  - ▶ one very detailed and close to the actual concurrent implementation
  - ▶ the other more abstract describing the abstract trees of relevant actions the systems has to perform
- Behavioral equivalences are used for assessing the conformance between the concrete implementation with respect to the abstract specification

# Process Algebras: System Verifications

Verification of concurrent system within the process algebraic approach is performed:

- by resorting to behavioural equivalences
- by checking that processes enjoy properties described by temporal logics formulae

## Model Checking

- Systems are specified as terms of a process description language
- Properties are specified as temporal logic formulae
- Model checking is used to determine whether the systems enjoy the property specified by the temporal formulae.

# Process Algebras: System Verifications

Verification of concurrent system within the process algebraic approach is performed:

- by resorting to behavioural equivalences
- by checking that processes enjoy properties described by temporal logics formulae

## Labelled Transition Systems

- In both approaches Labelled Transition Systems (LTS) play a crucial role
- LTS consist of:
  - ▶ a set of states, describing the configurations a system can reach
  - ▶ a set of transition labels, identifying the actions a system can perform to interact with the environment
  - ▶ a transition relation, denoting systems evolutions as determined by the execution of specific actions.

# Teaching Process Algebras

We aim at teaching the general theory of Process Algebras and seeing the different languages as specific instances of the general approach.

# Teaching Process Algebras

We aim at teaching the general theory of Process Algebras and seeing the different languages as specific instances of the general approach.

## PA Main Ingredients:

- 1 A minimal set of well thought operators aiming at capturing the relevant aspect of systems behavior and the way systems are composed.
- 2 A transition system associated to the algebra via structural *operational semantics* to describe the evolution of all systems that can be built from the operators.
- 3 An equivalence notion that permits abstracting from irrelevant details of systems descriptions.

# Teaching Process Algebras

We aim at teaching the general theory of Process Algebras and seeing the different languages as specific instances of the general approach.

## PA Main Ingredients:

- 1 A minimal set of well thought operators aiming at capturing the relevant aspect of systems behavior and the way systems are composed.
- 2 A transition system associated to the algebra via structural *operational semantics* to describe the evolution of all systems that can be built from the operators.
- 3 An equivalence notion that permits abstracting from irrelevant details of systems descriptions.

# Teaching Process Algebras

We aim at teaching the general theory of Process Algebras and seeing the different languages as specific instances of the general approach.

## PA Main Ingredients:

- 1 A minimal set of well thought operators aiming at capturing the relevant aspect of systems behavior and the way systems are composed.
- 2 A transition system associated to the algebra via structural *operational semantics* to describe the evolution of all systems that can be built from the operators.
- 3 An equivalence notion that permits abstracting from irrelevant details of systems descriptions.

# Teaching Process Algebras

We aim at teaching the general theory of Process Algebras and seeing the different languages as specific instances of the general approach.

## PA Main Ingredients:

- 1 A minimal set of well thought operators aiming at capturing the relevant aspect of systems behavior and the way systems are composed.
- 2 A transition system associated to the algebra via structural *operational semantics* to describe the evolution of all systems that can be built from the operators.
- 3 An equivalence notion that permits abstracting from irrelevant details of systems descriptions.

# Teaching Process Algebras

We aim at teaching the general theory of Process Algebras and seeing the different languages as specific instances of the general approach.

## Extras:

- 4 Abstract structures that are compositionally associated to terms to provide *denotational semantics*.
- 5 A set of laws (axioms) that characterize behavioural equivalences to obtain a so called *algebraic semantics*.

# Teaching Process Algebras

We aim at teaching the general theory of Process Algebras and seeing the different languages as specific instances of the general approach.

## Extras:

- 4 Abstract structures that are compositionally associated to terms to provide *denotational semantics*.
- 5 A set of laws (axioms) that characterize behavioural equivalences to obtain a so called *algebraic semantics*.

# Teaching Process Algebras

We aim at teaching the general theory of Process Algebras and seeing the different languages as specific instances of the general approach.

## Extras:

- ④ Abstract structures that are compositionally associated to terms to provide *denotational semantics*.
- ⑤ A set of laws (axioms) that characterize behavioural equivalences to obtain a so called *algebraic semantics*.

# TAPAs: A Tool for Teaching Process Algebras

- 1 Understanding the meaning of the different process algebras operators by showing how they can be used to compose terms and the changes induced on the composed transition systems.
- 2 Appreciating the close correspondence between terms and processes by consistently updating terms when the graphical representation of labelled transition systems is changed and redrawing process graphs when terms are modified.
- 3 Evaluating the different behavioral equivalences by having them on a single platform and checking the different equivalences by simply pushing different buttons.
- 4 Studying model checking via a user friendly tool that in case of failures provides appropriate counterexamples that help debugging the specification.

# TAPAs: A Tool for Teaching Process Algebras

- 1 Understanding the meaning of the different process algebras operators by showing how they can be used to compose terms and the changes induced on the composed transition systems.
- 2 Appreciating the close correspondence between terms and processes by consistently updating terms when the graphical representation of labelled transition systems is changed and redrawing process graphs when terms are modified.
- 3 Evaluating the different behavioral equivalences by having them on a single platform and checking the different equivalences by simply pushing different buttons.
- 4 Studying model checking via a user friendly tool that in case of failures provides appropriate counterexamples that help debugging the specification.

# TAPAs: A Tool for Teaching Process Algebras

- 1 Understanding the meaning of the different process algebras operators by showing how they can be used to compose terms and the changes induced on the composed transition systems.
- 2 Appreciating the close correspondence between terms and processes by consistently updating terms when the graphical representation of labelled transition systems is changed and redrawing process graphs when terms are modified.
- 3 Evaluating the different behavioral equivalences by having them on a single platform and checking the different equivalences by simply pushing different buttons.
- 4 Studying model checking via a user friendly tool that in case of failures provides appropriate counterexamples that help debugging the specification.

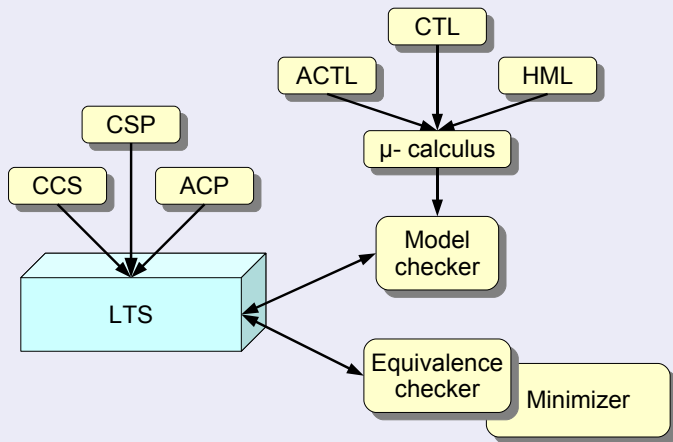
# TAPAs: A Tool for Teaching Process Algebras

- 1 Understanding the meaning of the different process algebras operators by showing how they can be used to compose terms and the changes induced on the composed transition systems.
- 2 Appreciating the close correspondence between terms and processes by consistently updating terms when the graphical representation of labelled transition systems is changed and redrawing process graphs when terms are modified.
- 3 Evaluating the different behavioral equivalences by having them on a single platform and checking the different equivalences by simply pushing different buttons.
- 4 Studying model checking via a user friendly tool that in case of failures provides appropriate counterexamples that help debugging the specification.

# TAPAs: A Tool for Teaching Process Algebras

- 1 Understanding the meaning of the different process algebras operators by showing how they can be used to compose terms and the changes induced on the composed transition systems.
- 2 Appreciating the close correspondence between terms and processes by consistently updating terms when the graphical representation of labelled transition systems is changed and redrawing process graphs when terms are modified.
- 3 Evaluating the different behavioral equivalences by having them on a single platform and checking the different equivalences by simply pushing different buttons.
- 4 Studying model checking via a user friendly tool that in case of failures provides appropriate counterexamples that help debugging the specification.

# TAPAs Architecture



# TAPAs: Processes and Process Systems

- In TAPAs concurrent systems are described by means of
  - ▶ *processes*, which are non-deterministic descriptions of system behaviours
  - ▶ *process systems*, which are obtained by process compositions
- Processes and process systems are composed by using the operators of a given process algebra.

For instance, in the case of CCS, a process system can be obtained by parallel composition, relabelling and restriction of processes.

# TAPAs: Processes and Process Systems

- In TAPAs concurrent systems are described by means of
  - ▶ *processes*, which are non-deterministic descriptions of system behaviours
  - ▶ *process systems*, which are obtained by process compositions
- Processes and process systems are composed by using the operators of a given process algebra.

For instance, in the case of CCS, a process system can be obtained by parallel composition, relabelling and restriction of processes.

# TAPAs: Processes and Process Systems

- In TAPAs concurrent systems are described by means of
  - ▶ *processes*, which are non-deterministic descriptions of system behaviours
  - ▶ *process systems*, which are obtained by process compositions
- Processes and process systems are composed by using the operators of a given process algebra.

For instance, in the case of CCS, a process system can be obtained by parallel composition, relabelling and restriction of processes.

# TAPAs: Processes and Process Systems

- In TAPAs concurrent systems are described by means of
  - ▶ *processes*, which are non-deterministic descriptions of system behaviours
  - ▶ *process systems*, which are obtained by process compositions
- Processes and process systems are composed by using the operators of a given process algebra.

For instance, in the case of CCS, a process system can be obtained by parallel composition, relabelling and restriction of processes.

# CCS Syntax

$T$	$::=$	(Processes)	$\text{let } D \text{ in } S$	(TAPAs CCS terms)		
	$\text{nil}$	(nil)	$D$	$::=$	(declarations)	
	$ $	$K$	(process name)	$K = T;$	(term decl.)	
	$ $	$\alpha.T$	(action prefix)	$ $	$D D$	(decl. list)
	$ $	$T + T$	(choice)	$S$	$::=$	(process systems)
	$ $	$T   T$	(parallel comp.)	$K$	(call)	
	$ $	$T[f]$	(relabelling)	$ $	$S   S$	(parallel comp.)
	$ $	$T \setminus C$	(restriction)	$ $	$S[f]$	(relabelling)
$f$	$::=$	(relabelling functions)	$ $	$S \setminus C$	(restriction)	
	$c/c$	(new / old)				
	$ $	$f, f$	(relabelling list)			
$\alpha$	$::=$	(actions)				
	$?c$	(input)				
	$ $	$!c$	(output)			
	$ $	$\text{tau}$	(silent action)			

# Textual and Graphical Representation of Processes

- TAPAs editor permits defining processes and system processes by using both graphical and textual representations
- A process is graphically represented by a graph whose edges are labeled with the actions it can perform.
- The same process can be represented (textually) by a term of a specific process algebra.
- A user can always change the process representation: TAPAs will guarantee the synchronization between the graph and the corresponding term.

TAPAs does not rely on a single process algebra that has to be used for the system specification. Indeed, even if currently only CCS has been considered, thanks to the modular implementation of TAPAs, other process algebras can be easily added.

# Textual and Graphical Representation of Processes

- TAPAs editor permits defining processes and system processes by using both graphical and textual representations
- A process is graphically represented by a graph whose edges are labeled with the actions it can perform.
- The same process can be represented (textually) by a term of a specific process algebra.
- A user can always change the process representation: TAPAs will guarantee the synchronization between the graph and the corresponding term.

TAPAs does not rely on a single process algebra that has to be used for the system specification. Indeed, even if currently only CCS has been considered, thanks to the modular implementation of TAPAs, other process algebras can be easily added.

# An Example: Bill and Ben

Textual specification:

*Bill* =?play.?meet.nil

*Ben* =?work.!meet.nil

# An Example: Bill and Ben

Textual specification:

*Bill* = ?play. ?meet. nil

*Ben* = ?work. !meet. nil

# An Example: Bill and Ben

Textual specification:

*Bill* = ?play. ?meet. nil

*Ben* = ?work. !meet. nil



# Bil and Ben: System Process

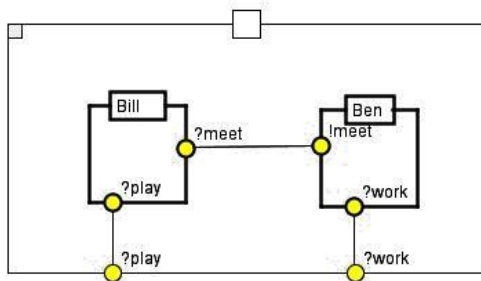
```
let  
  Bill = ?play.?meet.nil;  
  Ben = ?work.!meet.nil;  
in (Bill | Ben)\{meet};
```

## Bil and Ben: System Process

```
let  
  Bill = ?play.?meet.nil;  
  Ben = ?work.!meet.nil;  
in (Bill | Ben)\{meet};
```

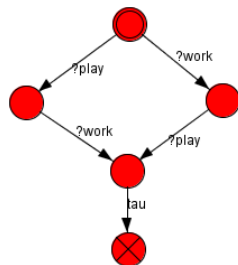
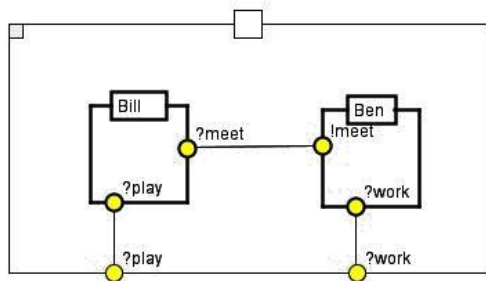
# Bil and Ben: System Process

```
let  
  Bill = ?play.?meet.nil;  
  Ben = ?work.!meet.nil;  
in (Bill | Ben)\{meet};
```



# Bil and Ben: System Process

```
let  
  Bill = ?play.?meet.nil;  
  Ben = ?work.!meet.nil;  
in (Bill | Ben)\{meet};
```



# TAPAs: Equivalence Checker

- The *Equivalences checker*, given two LTSs, permits verifying different kind of equivalences between them
  - ▶ It is worth noting that if other process algebras (e.g. value-passing CCS, CSP, ...) will be added to TAPAs, their integrations with the equivalence checker will be quite seamless.
- TAPAs considers two kind of equivalences:

– *Strong bisimulation (strong weak and branching)*

– *Weak bisimulation*

– *Simulation and co-simulation*

– *Ready simulation and co-simulation*

– *Ready simulation and co-simulation with ready actions*

– *Ready traces*

– *Ready traces*

# TAPAs: Equivalence Checker

- The *Equivalences checker*, given two LTSs, permits verifying different kind of equivalences between them
  - ▶ It is worth noting that if other process algebras (e.g. value-passing CCS, CSP, . . . ) will be added to TAPAs, their integrations with the equivalence checker will be quite seamless.
- TAPAs considers two kind of equivalences:
  - ▶ bsimulations (strong, weak and branching)

# TAPAs: Equivalence Checker

- The *Equivalences checker*, given two LTSs, permits verifying different kind of equivalences between them
  - ▶ It is worth noting that if other process algebras (e.g. value-passing CCS, CSP, . . . ) will be added to TAPAs, their integrations with the equivalence checker will be quite seamless.
- TAPAs considers two kind of equivalences:
  - ▶ bisimulations (strong, weak and branching)
  - ▶ decorated trace equivalences
    - ★ strong and weak trace;
    - ★ (weak) completed trace;
    - ★ divergence sensitive weak (completed) trace;
    - ★ must;
    - ★ testing;

# TAPAs: Equivalence Checker

- The *Equivalences checker*, given two LTSs, permits verifying different kind of equivalences between them
  - ▶ It is worth noting that if other process algebras (e.g. value-passing CCS, CSP, ...) will be added to TAPAs, their integrations with the equivalence checker will be quite seamless.
- TAPAs considers two kind of equivalences:
  - ▶ bisimulations (strong, weak and branching)
  - ▶ decorated trace equivalences
    - ★ strong and weak trace;
    - ★ (weak) completed trace;
    - ★ divergence sensitive weak (completed) trace;
    - ★ must;
    - ★ testing;

# TAPAs: Equivalence Checker

- The *Equivalences checker*, given two LTSs, permits verifying different kind of equivalences between them
  - ▶ It is worth noting that if other process algebras (e.g. value-passing CCS, CSP, . . . ) will be added to TAPAs, their integrations with the equivalence checker will be quite seamless.
- TAPAs considers two kind of equivalences:
  - ▶ bisimulations (strong, weak and branching)
  - ▶ decorated trace equivalences
    - ★ strong and weak trace;
    - ★ (weak) completed trace;
    - ★ divergence sensitive weak (completed) trace;
    - ★ must;
    - ★ testing;

# TAPAs: Model Checking

- TAPAs *model checker* takes in as an input only  $\mu$ -calculus formulae
- TAPAs can accept also other logics that permit specifying properties in more friendly ways
  - ▶ The formulae of these logics are translated in  $\mu$ -calculus formulae and the verifications are performed by using the algorithm for the model checking of  $\mu$ -calculus
- At the moment, besides the  $\mu$ -calculus, TAPAs accepts only one other logic: ACTL (Action Computation Tree Logic)
  - ▶ an action based version of the branching time logic CTL

# TAPAs: Model Checking

- TAPAs *model checker* takes in as an input only  $\mu$ -calculus formulae
- TAPAs can accept also other logics that permit specifying properties in more friendly ways
  - ▶ The formulae of these logics are translated in  $\mu$ -calculus formulae and the verifications are performed by using the algorithm for the model checking of  $\mu$ -calculus
- At the moment, besides the  $\mu$ -calculus, TAPAs accepts only one other logic: ACTL (Action Computation Tree Logic)
  - ▶ an action based version of the branching time logic CTL

# TAPAs: Model Checking

- TAPAs *model checker* takes in as an input only  $\mu$ -calculus formulae
- TAPAs can accept also other logics that permit specifying properties in more friendly ways
  - ▶ The formulae of these logics are translated in  $\mu$ -calculus formulae and the verifications are performed by using the algorithm for the model checking of  $\mu$ -calculus
- At the moment, besides the  $\mu$ -calculus, TAPAs accepts only one other logic: ACTL (Action Computation Tree Logic)
  - ▶ an action based version of the branching time logic CTL

# TAPAs: Model Checking

- TAPAs *model checker* takes in as an input only  $\mu$ -calculus formulae
- TAPAs can accept also other logics that permit specifying properties in more friendly ways
  - ▶ The formulae of these logics are translated in  $\mu$ -calculus formulae and the verifications are performed by using the algorithm for the model checking of  $\mu$ -calculus
- At the moment, besides the  $\mu$ -calculus, TAPAs accepts only one other logic: ACTL (Action Computation Tree Logic)
  - ▶ an action based version of the branching time logic CTL

# Mutual Exclusion Algorithm with TAPAs

Mutual exclusion algorithms are used in concurrent programming to avoid that pieces of code, called *critical sections*, simultaneously access a common resource, such as a shared variable. We consider the Peterson's algorithm, that allows two processes to share a single-use resource without conflict.

*P1*

```
while true do {  
  <noncritical section>  
  B1 = true;  
  K = 2;  
  while (B2 and K==2) do skip;  
  <critical section>  
  B1 = false;  
}
```

*P2*

```
while true do {  
  <noncritical section>  
  B2 = true;  
  K = 1;  
  while (B1 and K==1) do skip;  
  <critical section>  
  B2 = false;  
}
```

# Mutual Exclusion Algorithm with TAPAs

Mutual exclusion algorithms are used in concurrent programming to avoid that pieces of code, called *critical sections*, simultaneously access a common resource, such as a shared variable. We consider the Peterson's algorithm, that allows two processes to share a single-use resource without conflict.

*P1*

```
while true do {  
  <noncritical section>  
  B1 = true;  
  K = 2;  
  while (B2 and K==2) do skip;  
  <critical section>  
  B1 = false;  
}
```

*P2*

```
while true do {  
  <noncritical section>  
  B2 = true;  
  K = 1;  
  while (B1 and K==1) do skip;  
  <critical section>  
  B2 = false;  
}
```

## Mutual Exclusion Algorithm with TAPAs

Mutual exclusion algorithms are used in concurrent programming to avoid that pieces of code, called *critical sections*, simultaneously access a common resource, such as a shared variable. We consider the Peterson's algorithm, that allows two processes to share a single-use resource without conflict.

*P1*

```
while true do {  
  <noncritical section>  
  B1 = true;  
  K = 2;  
  while (B2 and K==2) do skip;  
  <critical section>  
  B1 = false;  
}
```

*P2*

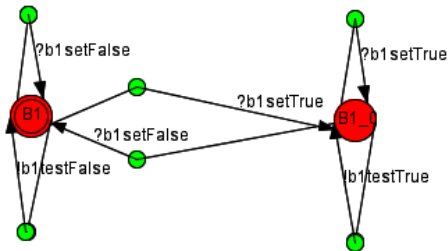
```
while true do {  
  <noncritical section>  
  B2 = true;  
  K = 1;  
  while (B1 and K==1) do skip;  
  <critical section>  
  B2 = false;  
}
```

# Mutual Exclusion Algorithm with TAPAs

- Variables  $B_1$ ,  $B_2$  and  $K$  can be modelled as two-states processes:
  - ▶ each state represents a value that the variable can assume

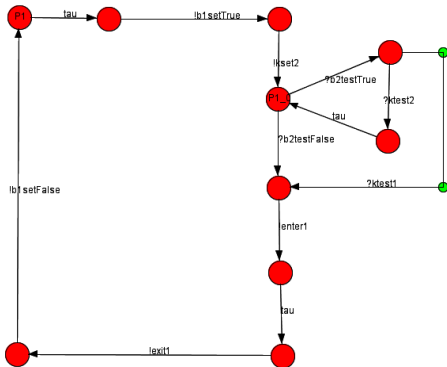
# Mutual Exclusion Algorithm with TAPAs

- Variables  $B_1$ ,  $B_2$  and  $K$  can be modelled as two-states processes:
  - ▶ each state represents a value that the variable can assume



# Mutual Exclusion Algorithm with TAPAs

- Processes  $P1$  and  $P2$  can be modelled follows:

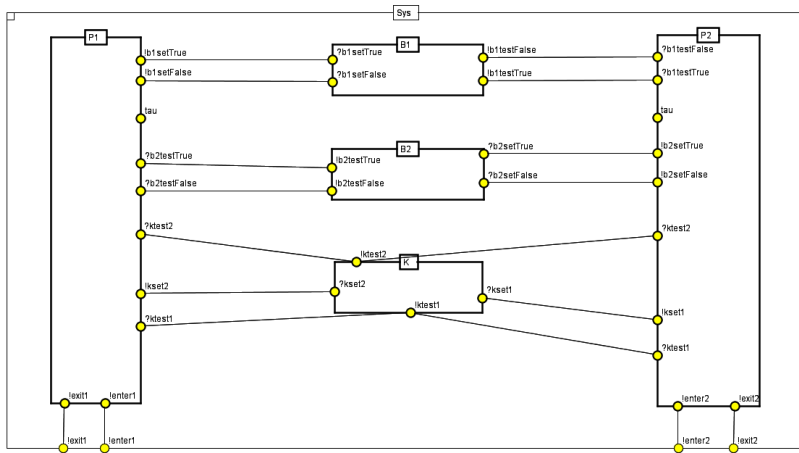


## Mutual Exclusion Algorithm with TAPAs

- The whole process system is created by putting the above five processes in parallel and by restricting the synchronization ports:

# Mutual Exclusion Algorithm with TAPAs

- The whole process system is created by putting the above five processes in parallel and by restricting the synchronization ports:

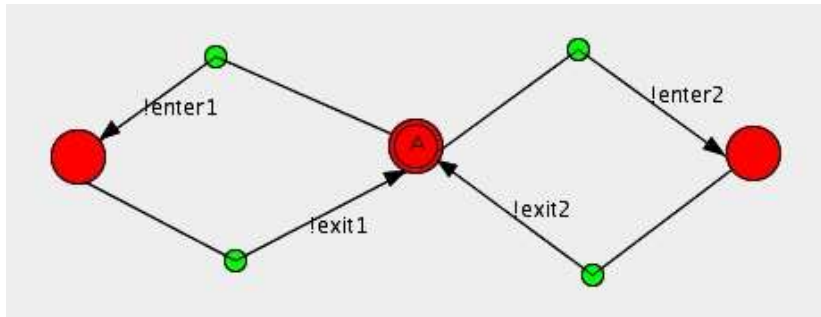


# Mutual Exclusion Algorithm with TAPAs

Using TAPAs we can verify the equivalences between the system's implementation and the following mutual exclusion specification:

## Mutual Exclusion Algorithm with TAPAs

Using TAPAs we can verify the equivalences between the system's implementation and the following mutual exclusion specification:



## Mutual Exclusion Algorithm with TAPAs

By using the tool a user can verify that the Peterson's algorithm enjoys the following relevant properties specified in  $\mu$ -calculus:

- deadlock-freedom: in each state, the system can perform at least an action

$$\nu X. \langle - \rangle \text{true} \wedge [-] X$$

- livelock-freedom: the system cannot reach a state where it can perform only infinite sequences of internal actions;

$$\neg \mu X. \langle - \rangle X \vee \nu Y. [-\tau] \text{false} \wedge \langle \tau \rangle \text{true} \wedge [\tau] Y$$

- starvation-freedom: if a process wants to enter its critical section, eventually it succeeds.

$$\mu X. [-] X \vee \langle !\text{enter } i \rangle \text{true}$$

# Conclusions

- We have introduced TAPAs, a tool for the specification and the analysis of concurrent systems.
  - ▶ TAPAs does not aim to compete with other existing tools
  - ▶ TAPAs is thought of as a teaching support for Theory of Concurrency lectures
  - ▶ The major feature that distinguish TAPAs from the other tools is its independence from a specific process algebra (and logic)
- The students that have developed simple (but realistic) case studies using TAPAs, have shown a deeper understanding of process algebras, behavioural equivalences and model checking.
- We plan to continue our development of the tool
  - ▶ by adding modules to deal with other process algebras
  - ▶ by allowing users to specify properties of systems by other logics
  - ▶ by adding other analysis tools (e.g. interactive execution)
- Thanks to TAPAs modularity, the different components can be developed independently.

# Conclusions

- We have introduced TAPAs, a tool for the specification and the analysis of concurrent systems.
  - ▶ TAPAs does not aim to compete with other existing tools
  - ▶ TAPAs is thought of as a teaching support for Theory of Concurrency lectures
  - ▶ The major feature that distinguish TAPAs from the other tools is its independence from a specific process algebra (and logic)
- The students that have developed simple (but realistic) case studies using TAPAs, have shown a deeper understanding of process algebras, behavioural equivalences and model checking.
- We plan to continue our development of the tool
  - ▶ by adding modules to deal with other process algebras
  - ▶ by allowing users to specify properties of systems by other logics
  - ▶ by adding other analysis tools (e.g. interactive execution)
- Thanks to TAPAs modularity, the different components can be developed independently.

# Conclusions

- We have introduced TAPAs, a tool for the specification and the analysis of concurrent systems.
  - ▶ TAPAs does not aim to compete with other existing tools
  - ▶ TAPAs is thought of as a teaching support for Theory of Concurrency lectures
  - ▶ The major feature that distinguish TAPAs from the other tools is its independence from a specific process algebra (and logic)
- The students that have developed simple (but realistic) case studies using TAPAs, have shown a deeper understanding of process algebras, behavioural equivalences and model checking.
- We plan to continue our development of the tool
  - ▶ by adding modules to deal with other process algebras
  - ▶ by allowing users to specify properties of systems by other logics
  - ▶ by adding other analysis tools (e.g. interactive execution)
- Thanks to TAPAs modularity, the different components can be developed independently.

# Conclusions

- We have introduced TAPAs, a tool for the specification and the analysis of concurrent systems.
  - ▶ TAPAs does not aim to compete with other existing tools
  - ▶ TAPAs is thought of as a teaching support for Theory of Concurrency lectures
  - ▶ The major feature that distinguish TAPAs from the other tools is its independence from a specific process algebra (and logic)
- The students that have developed simple (but realistic) case studies using TAPAs, have shown a deeper understanding of process algebras, behavioural equivalences and model checking.
- We plan to continue our development of the tool
  - ▶ by adding modules to deal with other process algebras
  - ▶ by allowing users to specify properties of systems by other logics
  - ▶ by adding other analysis tools (e.g. interactive execution)
- Thanks to TAPAs modularity, the different components can be developed independently.

# Conclusions

- We have introduced TAPAs, a tool for the specification and the analysis of concurrent systems.
  - ▶ TAPAs does not aim to compete with other existing tools
  - ▶ TAPAs is thought of as a teaching support for Theory of Concurrency lectures
  - ▶ The major feature that distinguish TAPAs from the other tools is its independence from a specific process algebra (and logic)
- The students that have developed simple (but realistic) case studies using TAPAs, have shown a deeper understanding of process algebras, behavioural equivalences and model checking.
- We plan to continue our development of the tool
  - ▶ by adding modules to deal with other process algebras
  - ▶ by allowing users to specify properties of systems by other logics
  - ▶ by adding other analysis tools (e.g. interactive execution)
- Thanks to TAPAs modularity, the different components can be developed independently.

Thank you for your attention!