Formal Modelling of Software Intensive Systems

TAPAs

Francesco Tiezzi

University of Camerino
francesco.tiezzi@unicam.it

A.A. 2015/2016

TAPAs

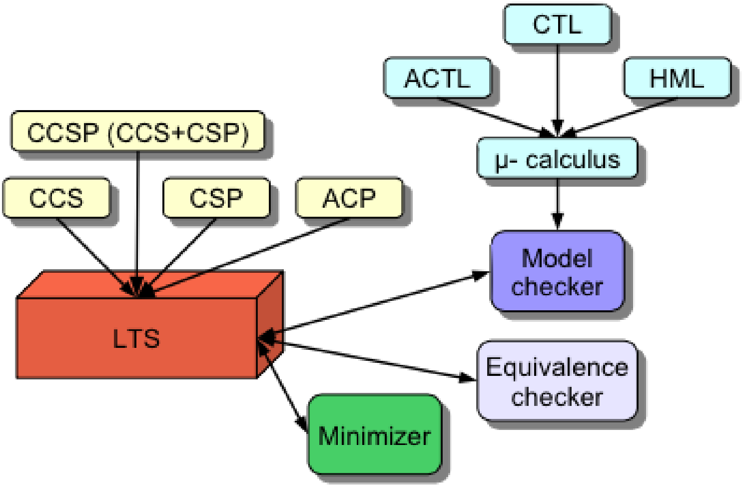
TAPAs

Software tool supporting teaching of process algebras

TAPAs permits:

- understanding the meaning of the different process algebras operators
- appreciating the close correspondence between **textual terms** and **graphical representations** of processes
- evaluating different behavioural equivalences
- model checking via a user-friendly tool that, in case of failures, provides appropriate **counterexamples**

TAPAs


CTL
ACTL
HML
CCSP (CCS+CSP)
μ- calculus
CCS
CSP
ACP
Model checker
LTS
Equivalence checker
Minimizer

TAPAs: CCSP (CCS+CSP) syntax

$$
\begin{array}{rcll}
\mathsf{M} & ::= & \mathsf{PROC\_DEC} \;\mid\; \mathsf{SYS\_DEC} \;\mid\; \mathsf{M}\;\mathsf{M} & \textit{(Module)} \\
\mathsf{PROC\_DEC} & ::= & \texttt{process P :} & \textit{(Process dec.)} \\
 & & \quad \mathtt{X}_1 = \sum_{i \in I_1} \mathsf{ACT}_1^i \,.\, \mathsf{PROC}_1^i & \\
 & & \quad \ldots & \\
 & & \quad \mathtt{X}_n = \sum_{j \in I_n} \mathsf{ACT}_n^j \,.\, \mathsf{PROC}_n^j & \\
 & & \texttt{end} & \\
\mathsf{ACT} & ::= & \texttt{tau} \;\mid\; \texttt{c!} \;\mid\; \texttt{c?} & \textit{(Action)} \\
\mathsf{PROC} & ::= & \texttt{nil} \;\mid\; \texttt{P[X]} \;\mid\; \texttt{S} & \textit{(Process)} \\
\mathsf{SYS\_DEC} & ::= & \texttt{system S :}\, \mathsf{COMP}\ \texttt{end} & \textit{(System dec.)} \\
\mathsf{COMP} & ::= & \mathsf{C} \;\mid\; \mathsf{C}_1\,(+)\,\mathsf{C}_2 \;\mid\; \mathsf{C}_1\,[\,]\,\mathsf{C}_2 \;\mid\; \mathsf{C}_1 \mid \mathsf{C}_2 & \textit{(Components)} \\
\mathsf{C} & ::= & \mathsf{PROC} & \textit{(Component)} \\
 & \mid & \texttt{sync on}\ \mathsf{CS}\ \texttt{in}\ \mathsf{C}_1 \mid \mathsf{C}_2\ \texttt{end} & \\
 & \mid & \texttt{rename [}\,\mathsf{F}\,\texttt{]}\ \texttt{in}\ \mathsf{COMP}\ \texttt{end} & \\
 & \mid & \texttt{restrict}\ \mathsf{CS}\ \texttt{in}\ \mathsf{COMP}\ \texttt{end} & \\
\mathsf{CS} & ::= & * \;\mid\; \{\mathtt{c}_1, \ldots, \mathtt{c}_n\} & \textit{(Channel set)} \\
\mathsf{F} & ::= & \texttt{c/c'} \;\mid\; \mathsf{F}, \mathsf{F} & \textit{(Renaming fun.)}
\end{array}
$$

TAPAs: CCSP (CCS+CSP) semantics 1/3

$$(P_{ref})\ \frac{(\texttt{process P : \ldots}\ \texttt{X}_i = \sum_{j\in I} \mathsf{ACT}_i^j\,.\,\mathsf{PROC}_i^j \ldots\ \textbf{end}) \in \mathsf{M}}{\texttt{P[X}_i\texttt{]} \xrightarrow{\mathsf{ACT}_i^k} \mathsf{PROC}_i^k}\ (k \in I)$$

$$(S_{ref})\quad \frac{(\texttt{system S} : \mathsf{COMP}\ \textbf{end}) \in \mathsf{M} \qquad \mathsf{COMP} \xrightarrow{\mu} \mathsf{COMP}'}{\mathsf{S} \xrightarrow{\mu} \mathsf{COMP}'}$$

TAPAs: CCSP (CCS+CSP) semantics 1/3

$$(P_{ref})\quad \frac{(\texttt{process P}\ :\ \ldots\ \texttt{X}_i =\ \sum_{j\in I} \mathsf{ACT}_i^j\,.\,\mathsf{PROC}_i^j \ldots\ \mathsf{end}) \in \mathsf{M}}{\texttt{P}[\texttt{X}_i] \xrightarrow{\mathsf{ACT}_i^k} \mathsf{PROC}_i^k}\ (k \in I)$$

$$(S_{ref})\quad \frac{(\texttt{system S}\ :\mathsf{COMP}\ \mathsf{end}) \in \mathsf{M} \qquad \mathsf{COMP} \xrightarrow{\mu} \mathsf{COMP}'}{\mathsf{S} \xrightarrow{\mu} \mathsf{COMP}'}$$

$$(Broad_1)\quad \frac{\mathsf{C}_1 \xrightarrow{\mu} \mathsf{C}' \qquad \mu \notin act(\mathsf{CS})}{\mathsf{sync\ on\ CS\ in\ C}_1 \mid \mathsf{C}_2\ \mathsf{end} \xrightarrow{\mu} \mathsf{sync\ on\ CS\ in\ C}' \mid \mathsf{C}_2\ \mathsf{end}}$$

$$(Broad_2)\quad \frac{\mathsf{C}_2 \xrightarrow{\mu} \mathsf{C}' \qquad \mu \notin act(\mathsf{CS})}{\mathsf{sync\ on\ CS\ in\ C}_1 \mid \mathsf{C}_2\ \mathsf{end} \xrightarrow{\mu} \mathsf{sync\ on\ CS\ in\ C}_1 \mid \mathsf{C}'\ \mathsf{end}}$$

$$(Broad_3)\quad \frac{\mathsf{C}_1 \xrightarrow{\alpha} \mathsf{C}_1' \qquad \mathsf{C}_2 \xrightarrow{\alpha} \mathsf{C}_2' \qquad \alpha \in act(\mathsf{CS})}{\mathsf{sync\ on\ CS\ in\ C}_1 \mid \mathsf{C}_2\ \mathsf{end} \xrightarrow{\alpha} \mathsf{sync\ on\ CS\ in\ C}_1' \mid \mathsf{C}_2'\ \mathsf{end}}$$

TAPAs: CCSP (CCS+CSP) semantics 2/3

$$(Ren)\quad \frac{\mathsf{COMP} \xrightarrow{\mu} \mathsf{COMP'}}{\texttt{rename}\,[\,\mathsf{F}\,]\,\texttt{in}\;\mathsf{COMP}\;\texttt{end} \xrightarrow{\mathsf{F}(\mu)} \texttt{rename}\,[\,\mathsf{F}\,]\,\texttt{in}\;\mathsf{COMP'}\;\texttt{end}}$$

$$(Res)\quad \frac{\mathsf{COMP} \xrightarrow{\mu} \mathsf{COMP'} \qquad \mu \notin act(\mathsf{CS})}{\texttt{restrict}\;\mathsf{CS}\;\texttt{in}\;\mathsf{COMP}\;\texttt{end} \xrightarrow{\mu} \texttt{restrict}\;\mathsf{CS}\;\texttt{in}\;\mathsf{COMP'}\;\texttt{end}}$$

TAPAs: CCSP (CCS+CSP) semantics 2/3

$$(Ren)\quad \frac{\mathsf{COMP} \xrightarrow{\mu} \mathsf{COMP}'}{\mathtt{rename}\ [\,\mathsf{F}\,]\ \mathtt{in}\ \mathsf{COMP}\ \mathtt{end} \xrightarrow{\mathsf{F}(\mu)} \mathtt{rename}\ [\,\mathsf{F}\,]\ \mathtt{in}\ \mathsf{COMP}'\ \mathtt{end}}$$

$$(Res)\quad \frac{\mathsf{COMP} \xrightarrow{\mu} \mathsf{COMP}' \qquad \mu \notin act(\mathsf{CS})}{\mathtt{restrict}\ \mathsf{CS}\ \mathtt{in}\ \mathsf{COMP}\ \mathtt{end} \xrightarrow{\mu} \mathtt{restrict}\ \mathsf{CS}\ \mathtt{in}\ \mathsf{COMP}'\ \mathtt{end}}$$

$$(Sync)\quad \frac{\mathsf{C}_1 \xrightarrow{\alpha} \mathsf{C}_1' \qquad \mathsf{C}_2 \xrightarrow{\bar{\alpha}} \mathsf{C}_2'}{\mathsf{C}_1 \mid \mathsf{C}_2 \xrightarrow{\mathtt{tau}} \mathsf{C}_1' \mid \mathsf{C}_2'}$$

$$(Inter_1)\quad \frac{\mathsf{C}_1 \xrightarrow{\mu} \mathsf{C}_1'}{\mathsf{C}_1 \mid \mathsf{C}_2 \xrightarrow{\mu} \mathsf{C}_1' \mid \mathsf{C}_2} \qquad\qquad (Inter_2)\quad \frac{\mathsf{C}_2 \xrightarrow{\mu} \mathsf{C}_2'}{\mathsf{C}_1 \mid \mathsf{C}_2 \xrightarrow{\mu} \mathsf{C}_1 \mid \mathsf{C}_2'}$$

TAPAs: CCSP (CCS+CSP) semantics 3/3

$$(Int.\ choice_1) \quad \mathsf{C}_1 (+) \mathsf{C}_2 \xrightarrow{\mathtt{tau}} \mathsf{C}_1 \qquad (Int.\ choice_2) \quad \mathsf{C}_1 (+) \mathsf{C}_2 \xrightarrow{\mathtt{tau}} \mathsf{C}_2$$

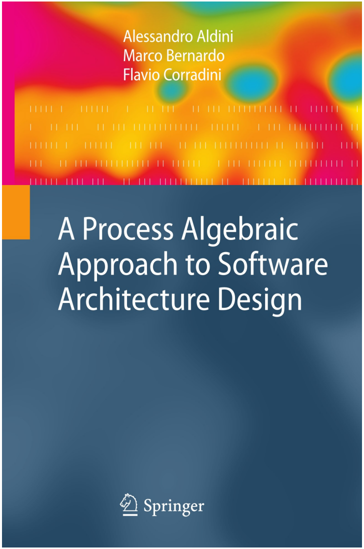
$$(Ext.\ choice_1) \quad \frac{\mathsf{C}_1 \xrightarrow{\alpha} \mathsf{C}'}{\mathsf{C}_1 [] \mathsf{C}_2 \xrightarrow{\alpha} \mathsf{C}'} \qquad (Ext.\ choice_2) \quad \frac{\mathsf{C}_2 \xrightarrow{\alpha} \mathsf{C}'}{\mathsf{C}_1 [] \mathsf{C}_2 \xrightarrow{\alpha} \mathsf{C}'}$$

$$(Ext.\ choice_3) \quad \frac{\mathsf{C}_1 \xrightarrow{\mathtt{tau}} \mathsf{C}'}{\mathsf{C}_1 [] \mathsf{C}_2 \xrightarrow{\mathtt{tau}} \mathsf{C}' [] \mathsf{C}_2} \qquad (Ext.\ choice_4) \quad \frac{\mathsf{C}_2 \xrightarrow{\mathtt{tau}} \mathsf{C}'}{\mathsf{C}_1 [] \mathsf{C}_2 \xrightarrow{\mathtt{tau}} \mathsf{C}_1 [] \mathsf{C}'}$$

Bill&Ben Example in TAPAs

Demo!

Producer-Consumer Example



Producer-Consumer Example

- The system is composed of
  - a producer
  - a finite-capacity buffer
  - a consumer
- The producer **deposits** items into the buffer as long as the buffer capacity is not exceeded
- Stored items can be **withdrawn** by the consumer according to some predefined discipline, like FIFO or LIFO
- Assumptions:
  - The buffer has only two positions
  - Items are all identical, so that the specific discipline that has been adopted for withdrawals is not important from the point of view of an external observer

Demo!

Producer-Consumer Example

- The system is composed of
  - a producer
  - a finite-capacity buffer
  - a consumer
- The producer **deposits** items into the buffer as long as the buffer capacity is not exceeded
- Stored items can be **withdrawn** by the consumer according to some predefined discipline, like FIFO or LIFO
- Assumptions:
  - The buffer has only two positions
  - Items are all identical, so that the specific discipline that has been adopted for withdrawals is not important from the point of view of an external observer

Demo!

Behavioural equivalences: bisimulation

Intuition (bisimulation game)

- Whenever a process can perform an action, then an equivalent process has to be able to respond with the same action
- Moreover, the corresponding derivative processes must still be equivalent to each other

Definition (bisimulation)

A binary relation $\mathcal{B}$ over processes is a bisimulation iff, whenever $(P_1, P_2) \in \mathcal{B}$, then for all actions a:

- Whenever $P_1 \xrightarrow{a} P_1'$, then $P_2 \xrightarrow{a} P_2'$ with $(P_1', P_2') \in \mathcal{B}$
- Whenever $P_2 \xrightarrow{a} P_2'$, then $P_1 \xrightarrow{a} P_1'$ with $(P_1', P_2') \in \mathcal{B}$

Behavioural equivalences: bisimulation

Definition (bisimilarity)

Bisimilarity, denoted $\sim_B$, is the union of all the bisimulations (which is the largest bisimulation)

Congruence

Bisimilarity is a *congruence* with respect to all the dynamic and static operators of CCS

Behavioural equivalences: weak bisimulation

Intuition (bisimulation game)

Whenever a process can perform an action, then an equivalent process has to be able to respond with the same action possibly preceded and followed by arbitrarily many τ-actions

Definition (weak bisimulation)

A binary relation $\mathcal{B}$ over processes is a weak bisimulation iff, whenever $(P_1, P_2) \in \mathcal{B}$, then:

- Whenever $P_1 \xrightarrow{\tau} P_1'$, then $P_2 \xRightarrow{\tau^*} P_2'$ with $(P_1', P_2') \in \mathcal{B}$
- Whenever $P_2 \xrightarrow{\tau} P_2'$, then $P_1 \xRightarrow{\tau^*} P_1'$ with $(P_1', P_2') \in \mathcal{B}$

and for all *visible* actions a:

- Whenever $P_1 \xrightarrow{a} P_1'$, then $P_2 \xRightarrow{\tau^* a \tau^*} P_2'$ with $(P_1', P_2') \in \mathcal{B}$
- Whenever $P_2 \xrightarrow{a} P_2'$, then $P_1 \xRightarrow{\tau^* a \tau^*} P_1'$ with $(P_1', P_2') \in \mathcal{B}$

Behavioural equivalences: weak bisimulation

Definition (bisimilarity)

Weak bisimilarity, denoted $\approx_B$, is the union of all the weak bisimulations
(which is the largest weak bisimulation)

Congruence

- Is weak bisimilarity a *congruence*?
- Consider $a.Nil$ and $\tau.a.Nil$, with the context $[\cdot] + b.Nil$

Definition (weak congruence)

P_1 is weakly bisimulation congruent to P_2 iff for all actions a:

- Whenever $P_1 \xrightarrow{a} P_1'$, then $P_2 \xRightarrow{\tau^* a \tau^*} P_2'$ with $P_1' \approx_B P_2'$
- Whenever $P_2 \xrightarrow{a} P_2'$, then $P_1 \xRightarrow{\tau^* a \tau^*} P_1'$ with $P_1' \approx_B P_2'$

Behavioural equivalences: weak bisimulation

Definition (bisimilarity)

Weak bisimilarity, denoted $\approx_B$, is the union of all the weak bisimulations
(which is the largest weak bisimulation)

Congruence

- Is weak bisimilarity a *congruence*?
- Consider $a.Nil$ and $\tau.a.Nil$, with the context $[\cdot] + b.Nil$

Definition (weak congruence)

P_1 is weakly bisimulation congruent to P_2 iff for all actions a:

- Whenever $P_1 \xrightarrow{a} P_1'$, then $P_2 \xRightarrow{\tau^* a \tau^*} P_2'$ with $P_1' \approx_B P_2'$
- Whenever $P_2 \xrightarrow{a} P_2'$, then $P_1 \xRightarrow{\tau^* a \tau^*} P_1'$ with $P_1' \approx_B P_2'$

Producer-Consumer Example in TAPAs

- The system is composed of
  - a producer
  - a finite-capacity buffer
  - a consumer
- The producer **deposits** items into the buffer as long as the buffer capacity is not exceeded
- Stored items can be **withdrawn** by the consumer according to some predefined discipline, like FIFO or LIFO
- Assumptions:
  - The buffer has only two positions
  - Items are all identical, so that the specific discipline that has been adopted for withdrawals is not important from the point of view of an external observer

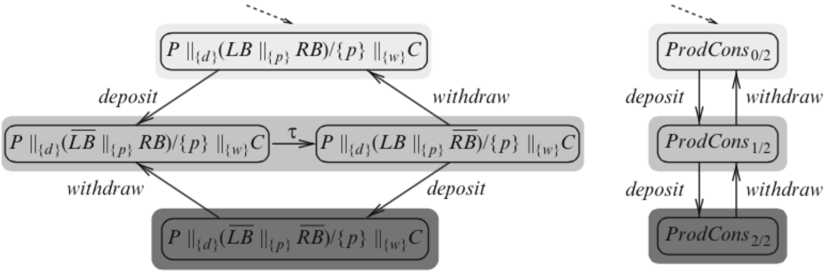
Demo!

Producer-Consumer Example in TAPAs

- The system is composed of
  - a producer
  - a finite-capacity buffer
  - a consumer
- The producer **deposits** items into the buffer as long as the buffer capacity is not exceeded
- Stored items can be **withdrawn** by the consumer according to some predefined discipline, like FIFO or LIFO
- Assumptions:
  - The buffer has only two positions
  - Items are all identical, so that the specific discipline that has been adopted for withdrawals is not important from the point of view of an external observer

Demo!

Producer-Consumer Example in TAPAs



Behavioural equivalences: trace and testing

Trace equivalence

Trace equivalence relates two process terms whenever they are able to execute the same sequences of visible actions

Testing equivalence

Testing equivalence relates two process terms whenever an external observer is not able to distinguish between them by interacting with them by means of tests and comparing their reactions

- *Test*: interaction with the process under test by means of a parallel process; the test is passed when the success action ω is executed
- *Testing equivalence* is the intersection of two behavioral equivalences, which are the kernels of two preorders related to the possibility (*may*) and the necessity (*must*) of passing tests

Producer-Consumer Example in TAPAs

- The system is composed of
  - a producer
  - a finite-capacity buffer
  - a consumer
- The producer **deposits** items into the buffer as long as the buffer capacity is not exceeded
- Stored items can be **withdrawn** by the consumer according to some predefined discipline, like FIFO or LIFO
- Assumptions:
  - The buffer has only two positions
  - Items are all identical, so that the specific discipline that has been adopted for withdrawals is not important from the point of view of an external observer

Demo!

Model checking: ACTL syntax

Action-based Computational Tree Logic (ACTL)

Propositional branching-time temporal logic interpreted over LTSs

Syntax:

State formulas	$\phi ::= \mathrm{tt} \mid \mathrm{ff} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \exists\,\gamma \mid \forall\gamma$
Path formulas	$\gamma ::= X_\chi\phi \mid X_\tau\phi \mid \phi\,{}_\chi U_{\chi'}\,\phi' \mid \phi\,{}_\chi U\,\phi' \mid F\,\phi \mid G\,\phi$
Action formulas	$\chi ::= \alpha \mid \neg\chi \mid \chi \wedge \chi' \mid \chi \vee \chi'$

Model checking: ACTL semantics

Action-based Computational Tree Logic (ACTL)

Propositional branching-time temporal logic interpreted over LTSs

Semantics:

$q \models tt$	for all $q \in S$.
$q \models \neg\phi$	iif $q \not\models \phi$.
$q \models \phi \vee \phi'$	iif $q \models \phi$ or $q \models \phi'$.
$q \models \exists\gamma$	iif $\exists\, \rho \in path(q)$ such that $\rho \models \gamma$.
$q \models \forall\gamma$	iif $\forall\, \rho \in path(q)$ such that $\rho \models \gamma$.
$\rho \models X_\chi\phi$	iif $\rho = (q, \alpha, q')\theta$, $q' \models \phi$ and $\alpha \models \chi$.
$\rho \models X_\tau\phi$	iif $\rho = (q, \tau, q')\theta$ and $q' \models \phi$.
$\rho \models \phi_\chi U_{\chi'}\phi'$	iif exists $\theta = (q, \alpha, q')\theta'$ suffix of ρ, such that $q \models \phi$, $\alpha \models \chi'$, $q' \models \phi'$ and for all suffixes $\theta_1 = (q_1, \beta, q_1')$ of ρ, such that θ is a proper suffix of θ_1, then $q_1 \models \phi$ and $\beta \models \chi$ or $\beta = \tau$.
$\rho \models \phi_\chi U\phi'$	iif exists $\theta = (q, \alpha, q')\theta'$ suffix of ρ, such that $q \models \phi$ and for all suffixes $\theta_1 = (q_1, \beta, q_1')$ of ρ, such that θ is a proper suffix of θ_1, then $q_1 \models \phi$ and $\beta \models \chi$ or $\beta = \tau$.
$\alpha \models \beta$	iif $\alpha = \beta$.
$\alpha \models \neg\chi$	iif $\alpha \not\models \chi$.
$\alpha \models \chi \vee \chi'$	iif $\alpha \models \chi \vee \alpha \models \chi'$.

Producer-Consumer Example in TAPAs

- The system is composed of
  - a producer
  - a finite-capacity buffer
  - a consumer
- The producer **deposits** items into the buffer as long as the buffer capacity is not exceeded
- Stored items can be **withdrawn** by the consumer according to some predefined discipline, like FIFO or LIFO
- Assumptions:
  - The buffer has only two positions
  - Items are all identical, so that the specific discipline that has been adopted for withdrawals is not important from the point of view of an external observer

Demo!