ObsGraph : a Tool for Modular Verification of Inter-enterprise Business Processes

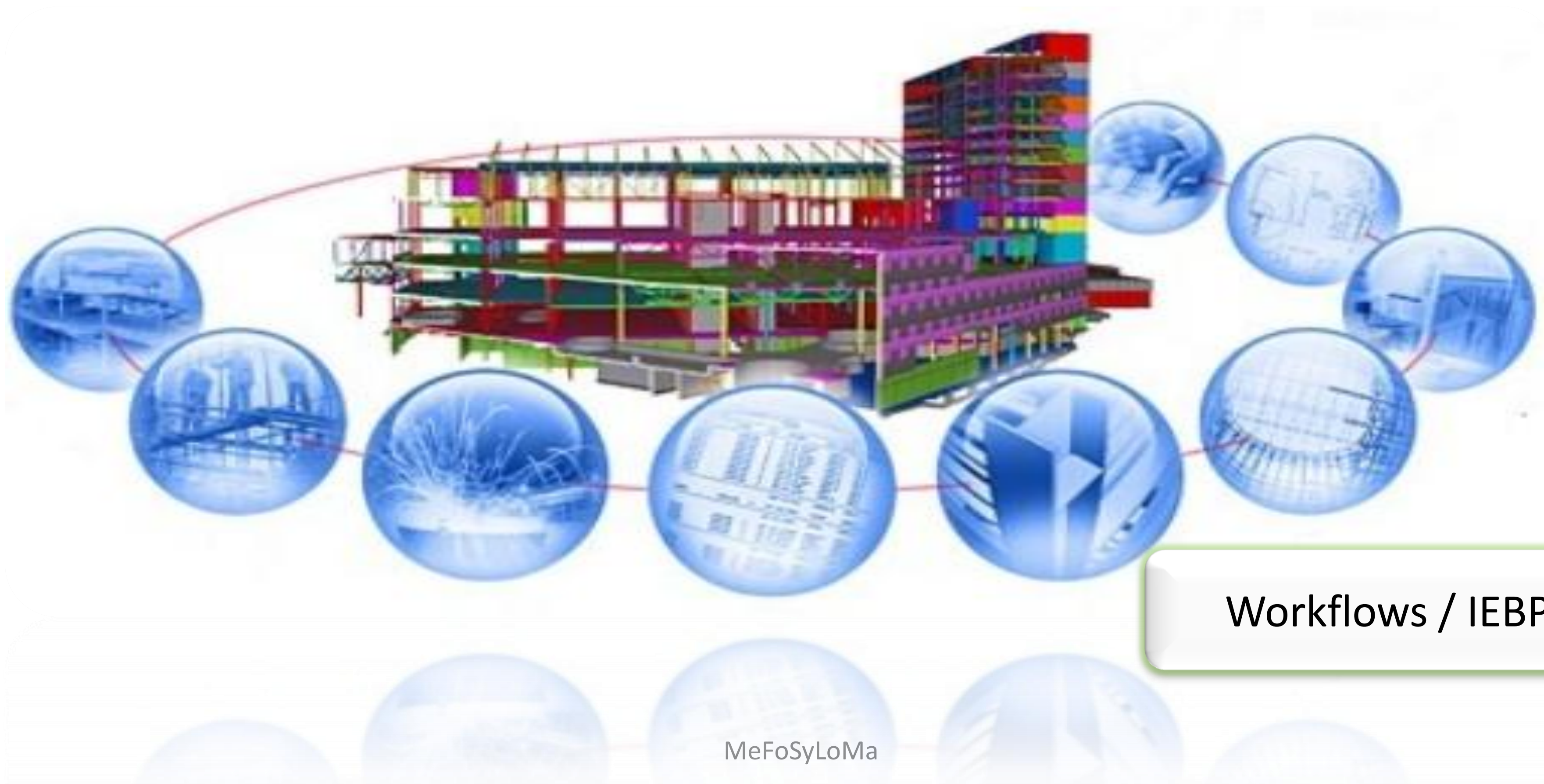
Hanen OCHI

Kais KLAI

LIPN – MeFoSyLoMa - Juin 2012

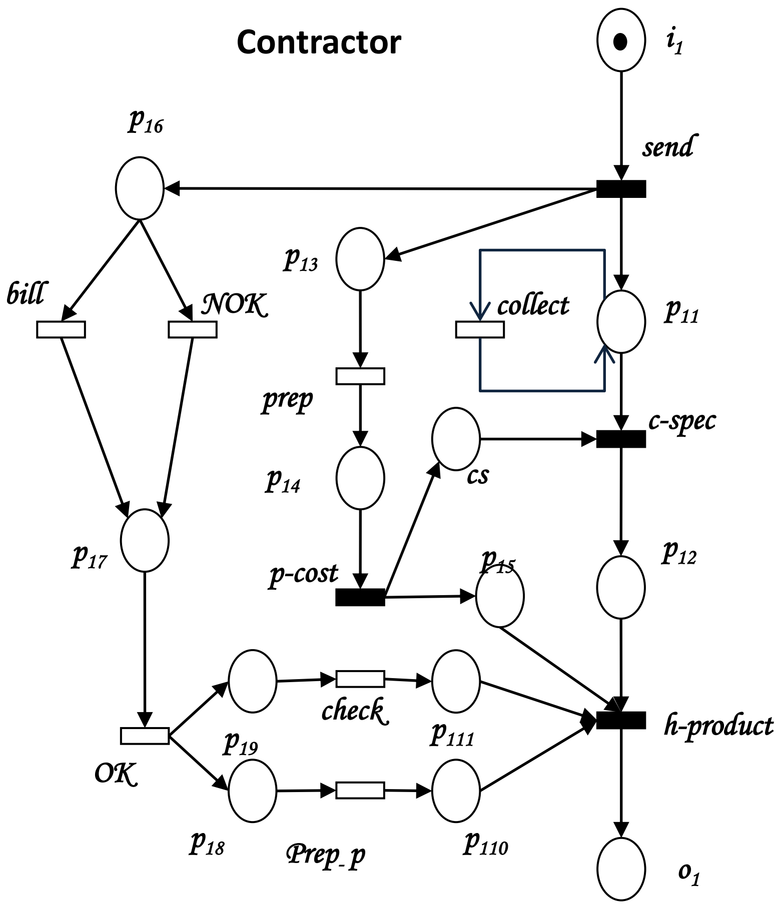
- Introduction
- Related work
- Abstraction and Verification of Inter-enterprise Business Processes (IEBP)
- Experimental results
- Conclusion

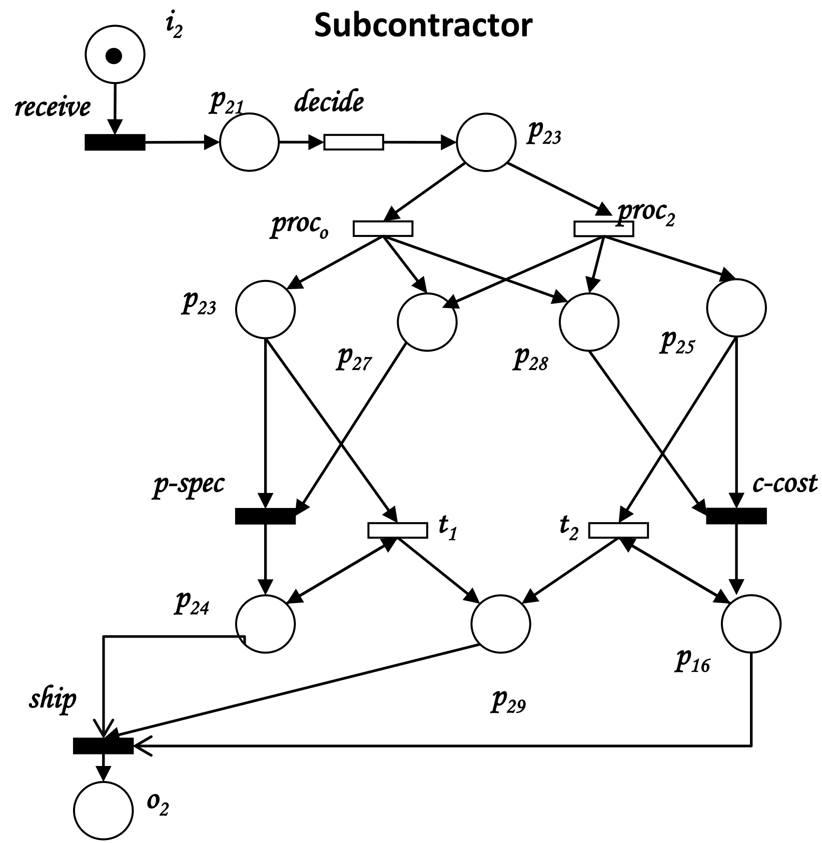
Motivation



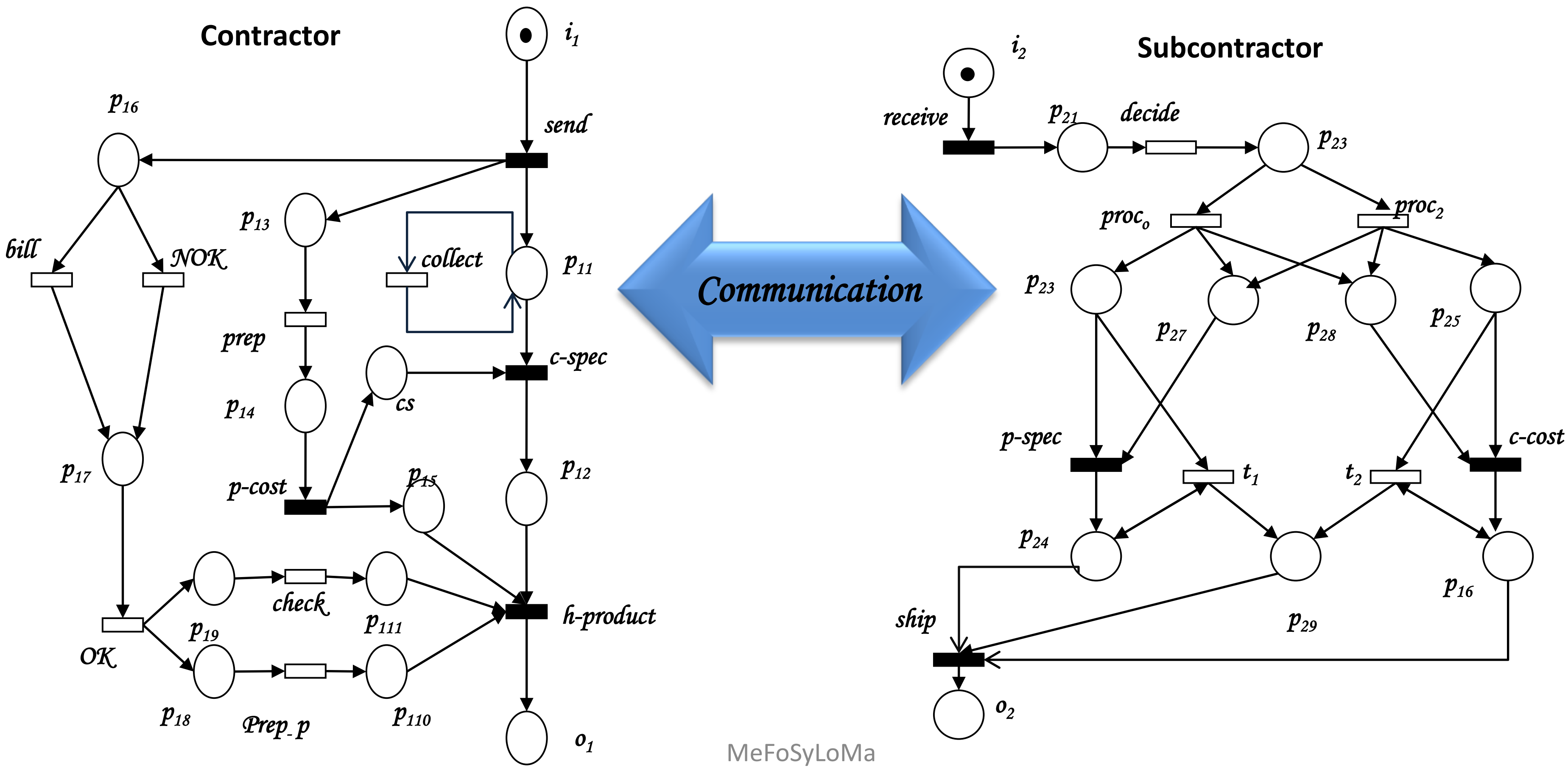
Workflows / IEBP

Motivation


Contractor
i1
send
p16
p13
p11
bill
NOK
collect
prep
c-spec
p14
cs
p17
p-cost
p15
p12
check
p111
h-product
OK
p19
p18
Prep_p
p110
o1


i2
Subcontractor
receive
p21
decide
p23
proco
proc2
p23
p27
p28
p25
p-spec
c-cost
t1
t2
p24
p16
ship
p29
o2

Motivation


Contractor
i_1
send
p_{16}
p_{13}
collect
p_{11}
bill
NOK
prep
c-spec
p_{14}
cs
p_{17}
p-cost
p_{15}
p_{12}
check
p_{111}
h-product
OK
p_{19}
p_{18}
Prep_p
p_{110}
o_1
Communication
Subcontractor
i_2
receive
p_{21}
decide
p_{23}
$proc_o$
$proc_2$
p_{23}
p_{27}
p_{28}
p_{25}
p-spec
t_1
t_2
c-cost
p_{24}
p_{16}
p_{29}
ship
o_2

Motivation

Motivation

Motivation

Related work

Model checking approches:

Explicit approaches:

Abstraction : States are represented by the graph's nodes

Related work

Model checking approches:

Explicit approaches:
Abstraction : States are represented by the graph's nodes

Symbolic approaches :
Abstraction : States are represented by BDD techniques

Related work

Model checking approches:

Explicit approaches:
<u>*Abstraction :*</u> States are represented by the graph's nodes

Symbolic approaches :
<u>*Abstraction :*</u> States are represented by BDD techniques

Hybrid approaches:
<u>*Abstraction :*</u> Graph's nodes representing a set of states are encoded using BDD techniques + the graph is represented explicitly

Verification of IEBP: Explicit approaches

- **Operating Guideline**
  - ✓ Abstraction used on SOA for services
  - ✓ Annotated automata
  - ✓ Verification of constraints represented as nodes' annotations

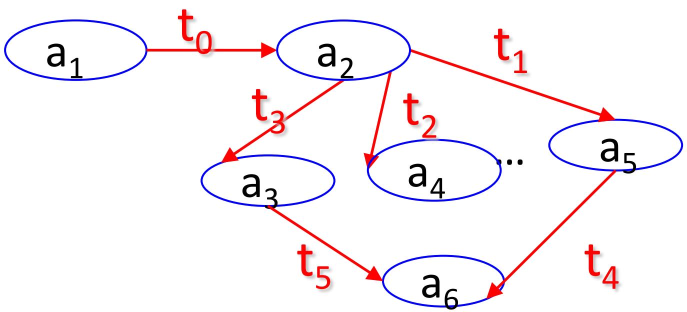
- **Communication graph**
  - ✓ Abstraction used for web services
  - ✓ A bi-part graph: visible nodes + hidden nodes
  - ✓ Verification of graph's paths

Hybrid approaches:

- **Symbolic Observation Graph SOG**
  - ✓ Abstraction of the reachability graph
  - ✓ Model checking
  - ✓ Events occurring in the formula: ***Obs***
  - ✓ Events not occurring in the formula: ***UnObs***
  - ✓ Structure :

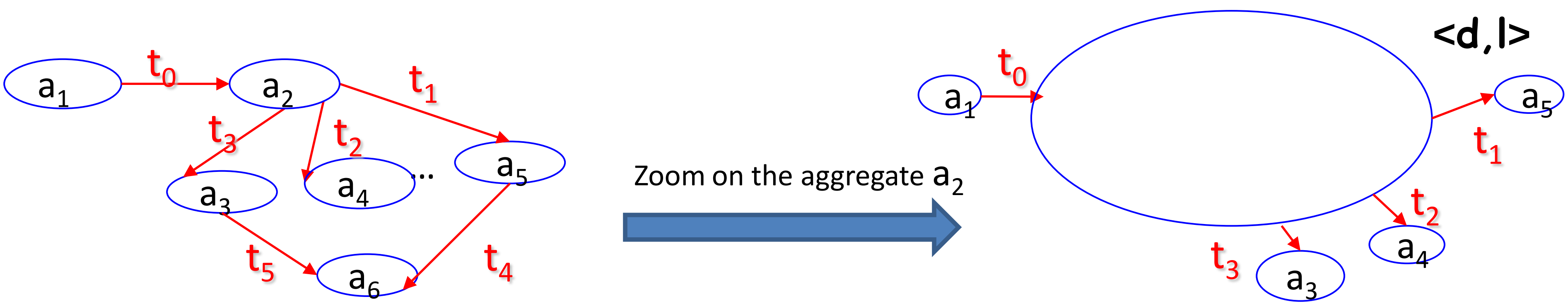
Hybrid approaches:

- **Symbolic Observation Graph SOG**
  - ✓ Abstraction of the reachability graph
  - ✓ Model checking
  - ✓ Events occurring in the formula: ***Obs***
  - ✓ Events not occurring in the formula: ***UnObs***
  - ✓ Structure :



Hybrid approaches:

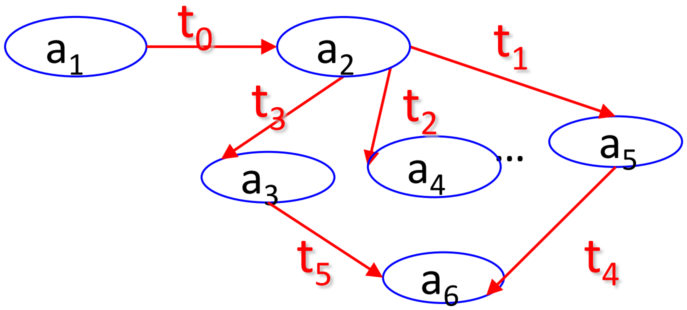
- **Symbolic Observation Graph SOG**
  - ✓ Abstraction of the reachability graph
  - ✓ Model checking
  - ✓ Events occurring in the formula: ***Obs***
  - ✓ Events not occurring in the formula: ***UnObs***
  - ✓ Structure :



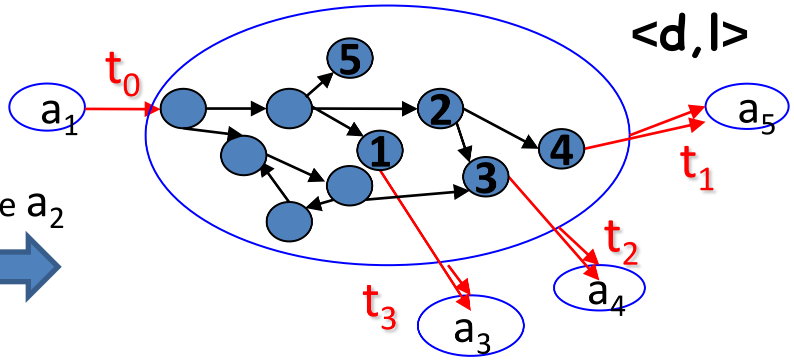
Hybrid approaches:

- **Symbolic Observation Graph SOG**
  - ✓ Abstraction of the reachability graph
  - ✓ Model checking
  - ✓ Events occurring in the formula: ***Obs***
  - ✓ Events not occurring in the formula: ***UnObs***
  - ✓ Structure :

    Node : Set of states linked by *unobserved* actions



Zoom on the aggregate a_2

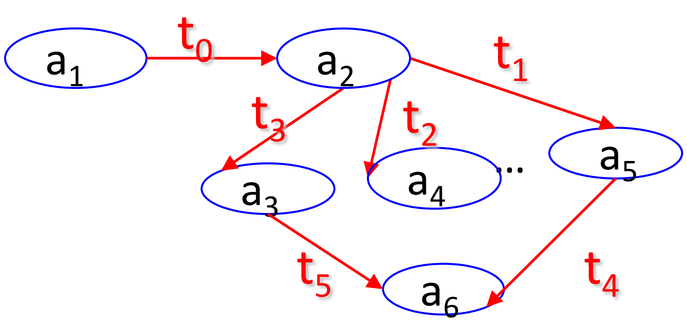


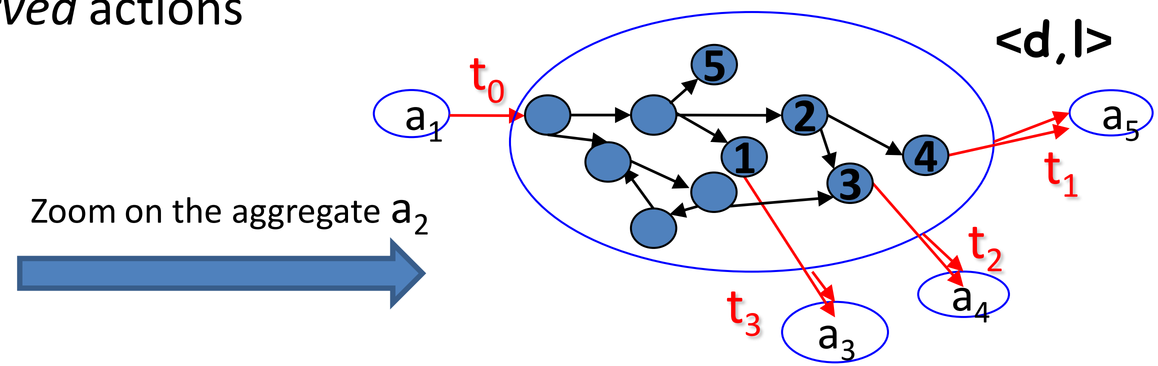
Hybrid approaches:

- **Symbolic Observation Graph SOG**
  - ✓ Abstraction of the reachability graph
  - ✓ Model checking
  - ✓ Events occurring in the formula: ***Obs***
  - ✓ Events not occurring in the formula: ***UnObs***
  - ✓ Structure :

    Node : Set of states linked by *unobserved* actions

    Edges : labled by *observed* actions





Abstraction

- **New version of Symbolic Observation Graph (SOG) for a workflow** :

  ✓Observation of only collaborative actions

  ✓Adding {*term*} : additional virtual observed action for proper termination
  (Act=Obs U UnObs U {*term*})

  ✓ Terminal circuit ⇔ deadlock state

  ✓ Observed behavor : λ

  => Nodes : Aggregates <S, λ>

Abstraction

•Comportement Observé <λ>

Définitions

1. $\lambda_{\mathcal{T}} : \mathcal{T} \rightarrow 2^{Obs}$

$$\lambda_{\mathcal{T}}(s) = (Enable(Sat(s)) \cap Obs) \cup \{term\} \text{ si } F \cap Sat(s) \neq \emptyset$$
$$(Enable(Sat(s)) \cap Obs) \cup \{term\} \text{ sinon}$$

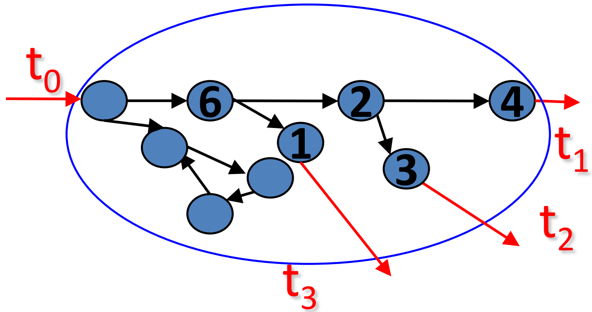
2 . $\lambda_{\mathcal{T}} : 2 \rightarrow 2^{Obs}$

$$\lambda_{\mathcal{T}}(S) = \{\lambda_{\mathcal{T}}(m) \mid m \in S\}$$

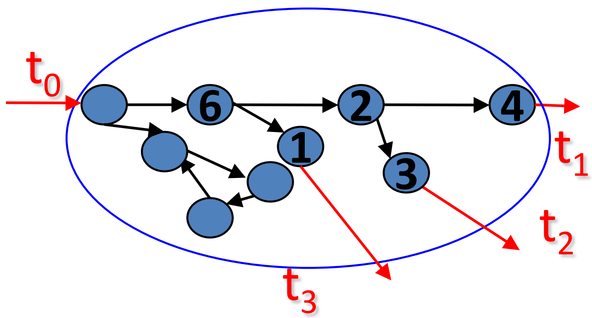
3. $\lambda_{min} : 2 \rightarrow 2^{2Obs}$

$$\lambda_{min}(S) = \{X \in \lambda_{\mathcal{T}}(S) \mid \nexists Y \in \lambda_{\mathcal{T}}(S) : Y \subset (X \setminus \{term\})\}$$

Observed behavior

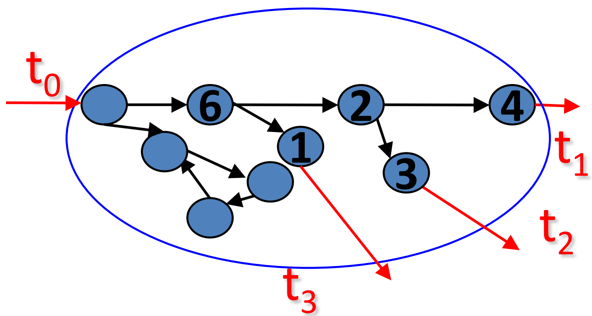


Observed behavior



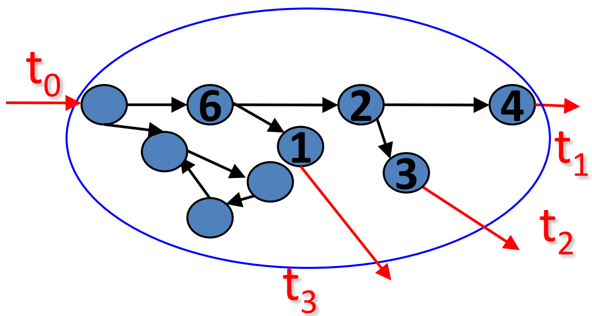
$\lambda = \{\{t_1\},\{t_2\},\{t_3\},\{t_1, t_2\},\{t_1, t_2, t_3\},\{\emptyset\}\}$

Observed behavior



$\lambda = \{\{t_1\},\{t_2\},\{t_3\},\{t_1, t_2\},\{t_1, t_2,t_3\},\{\emptyset\}\}$

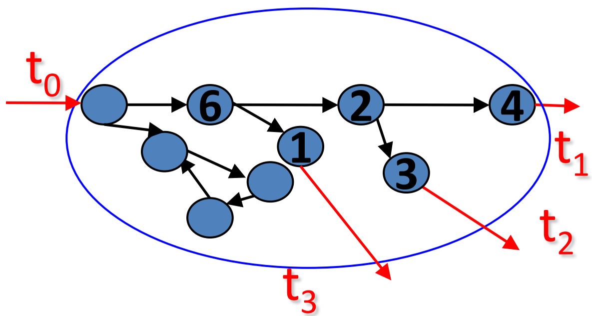
Observed behavior



$\lambda = \{\{t_1\},\{t_2\},\{t_3\},\{t_1, t_2\},\{t_1, t_2, t_3\},\{\emptyset\}\}$

$\Rightarrow \lambda = \{\{t_1\},\{t_2\},\{t_3\},\{\emptyset\}\}$

Observed behavior



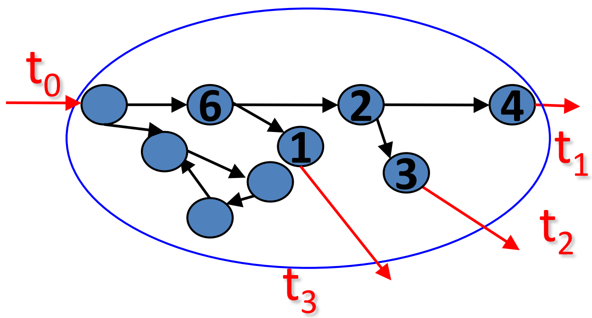
$\lambda = \{\{t_1\},\{t_2\},\{t_3\},\{t_1, t_2\},\{t_1, t_2,t_3\},\{\emptyset\}\}$

$\Rightarrow \lambda = \{\{t_1\},\{t_2\},\{t_3\},\{\emptyset\}\}$

- **Theorem** : Deadlock freeness

  A SOG $\mathcal{G}$ is said to be deadlock free $\Leftrightarrow \nexists\, a \in \mathcal{G} \mid \emptyset \in a.\lambda$

Observed behavior



$\lambda = \{\{t_1\},\{t_2\},\{t_3\},\{t_1, t_2\},\{t_1, t_2, t_3\},\{\emptyset\}\}$

$\Rightarrow \lambda = \{\{t_1\},\{t_2\},\{t_3\},\{\emptyset\}\}$

- **Theorem** : Deadlock freeness

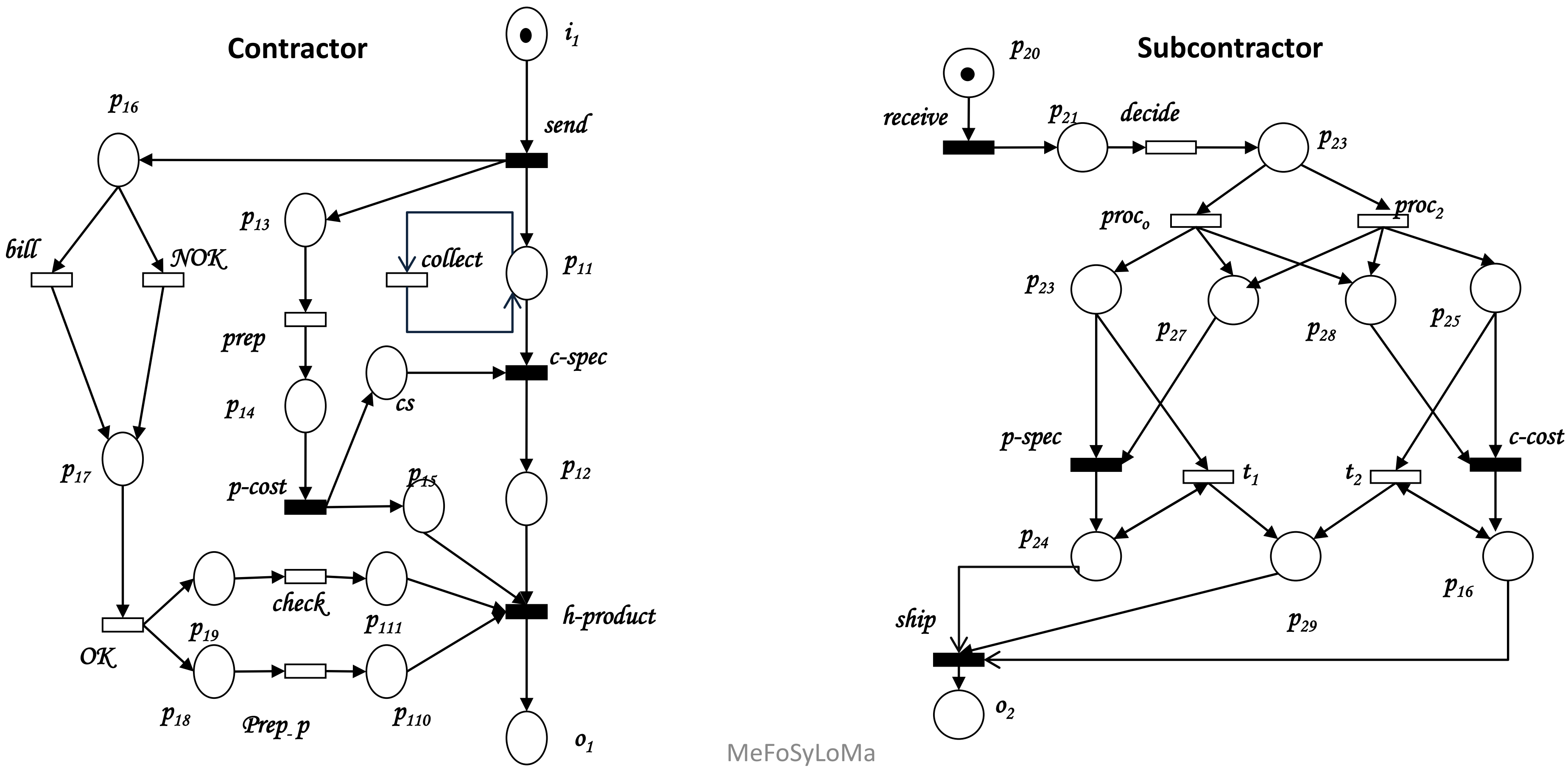
  A SOG $\mathcal{G}$ is said to be deadlock free $\Leftrightarrow \nexists\, a \in \mathcal{G} \mid \emptyset \in a.\lambda$

- **Proposition** :

  Let $\mathcal{WF}$ a BP and let $\mathcal{G}$ the asociated SOG

  $\mathcal{WF}$ has a deadlock state $\Leftrightarrow \exists\, a \in \mathcal{G} \mid \emptyset \in a.\lambda$

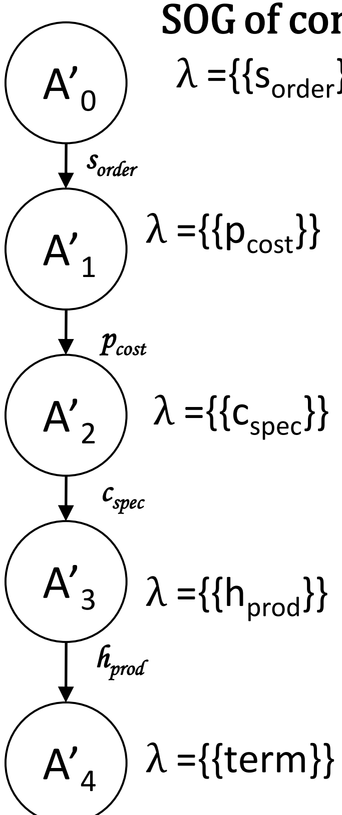
Example



Contractor
i1
send
p16
p13
bill
NOK
collect
p11
prep
c-spec
cs
p14
p17
p-cost
p15
p12
check
h-product
p111
p19
OK
p18
Prep_p
p110
o1
Subcontractor
p20
receive
p21
decide
p23
proc0
proc2
p23
p27
p28
p25
p-spec
c-cost
t1
t2
p24
p16
ship
p29
o2

Example (SOGs)

SOG of contractor



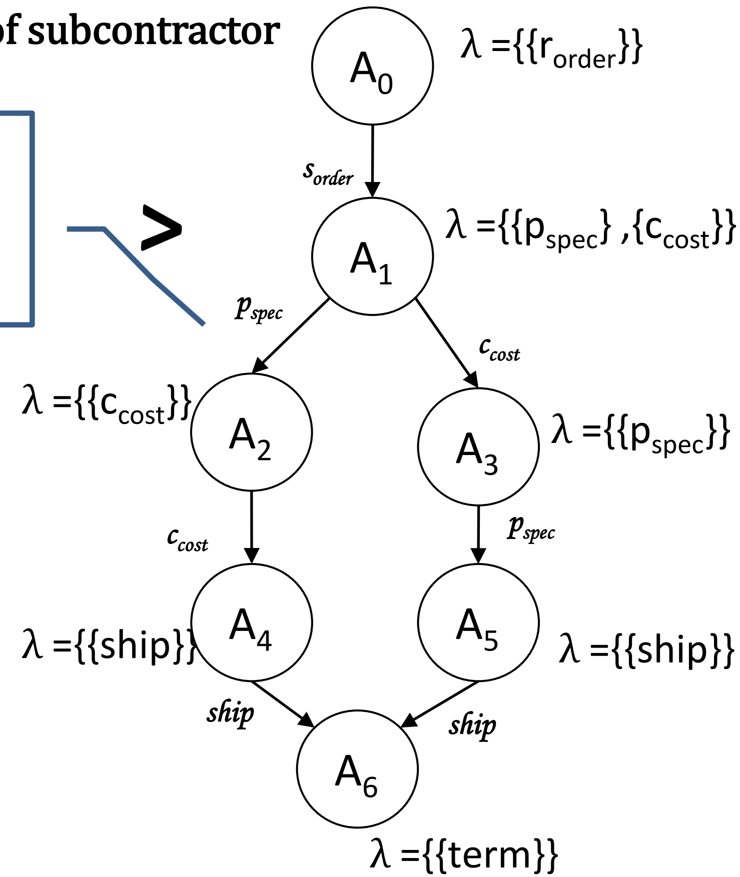
<

Reachability graph : 26 nodes + 66 edges

Reachability graph : 21 nodes + 22 edges

>

SOG of subcontractor



Composition of SOGs

- Locally $a = \langle S, \lambda \rangle$

Composition of SOGs

- Locally $a = <S, \lambda>$ **For composition** $\Rightarrow$ $a = <\lambda>$

Composition of SOGs

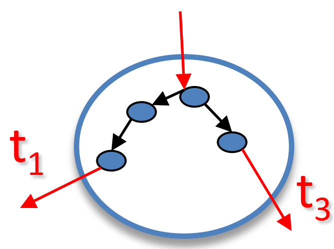
- Locally $a = <S, \lambda>$ **For composition** $\Rightarrow$ $a = <\lambda>$

- Synchronized product of two (or more) SOGs :
  Compute the observed behavior of $a = a1 \times a2$

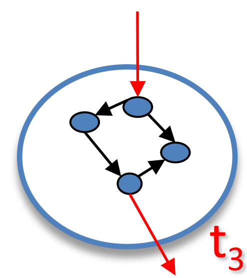
Composition of SOGs

- Locally $a = <S, \lambda>$ **For composition** $a = <\lambda>$

- Synchronized product of two (or more) SOGs :
  Compute the observed behavior of a= a1x a2



$\lambda_1 = \{\{t_1\}, \{t_3\}\}$

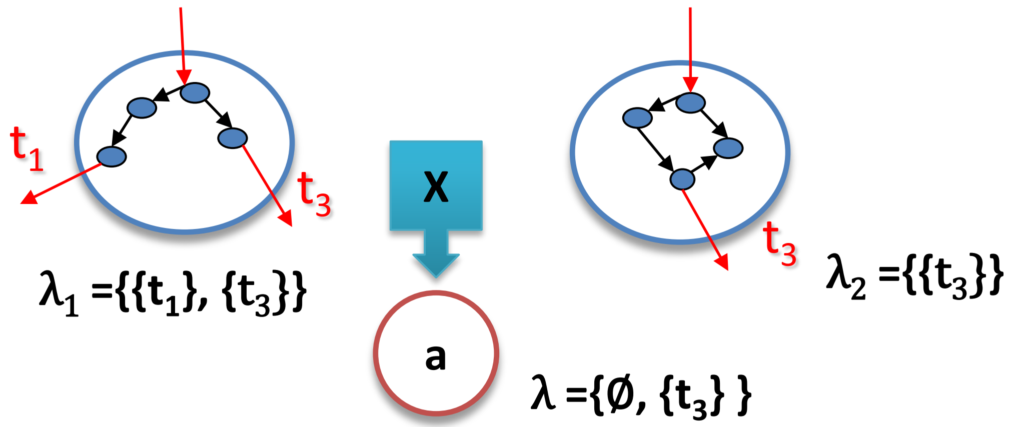


$\lambda_2 = \{\{t_3\}\}$

Composition of SOGs

- Locally $a = <S, \lambda>$ **For composition** ⇒ $a = <\lambda>$

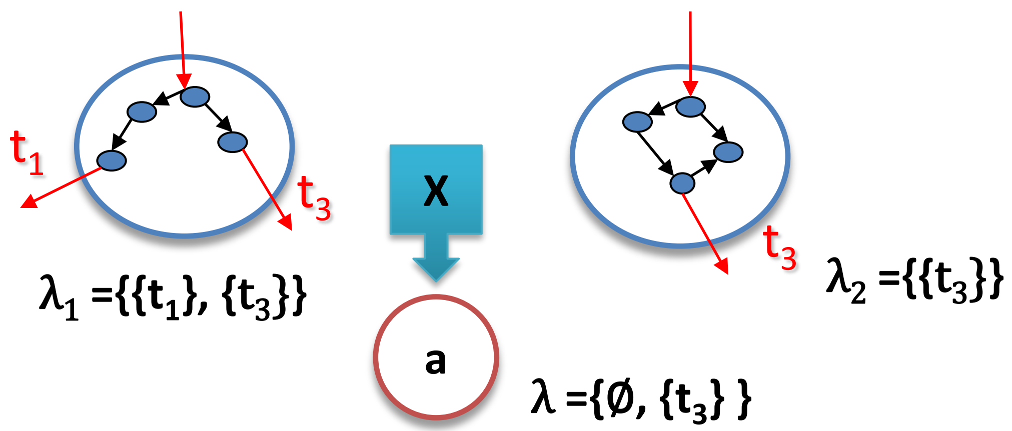
- Synchronized product of two (or more) SOGs **:**
  Compute the observed behavior of a= a1x a2



Composition of SOGs

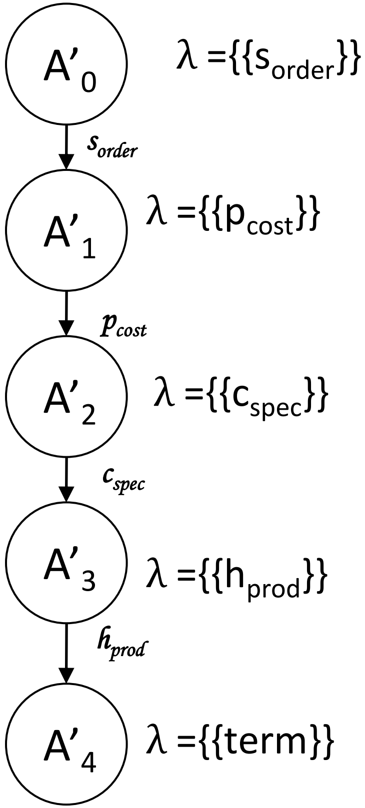
- Locally $a = \langle S, \lambda \rangle$ **For composition** $a = \langle \lambda \rangle$

- Synchronized product of two (or more) SOGs :
  Compute the observed behavior of a= a1x a2

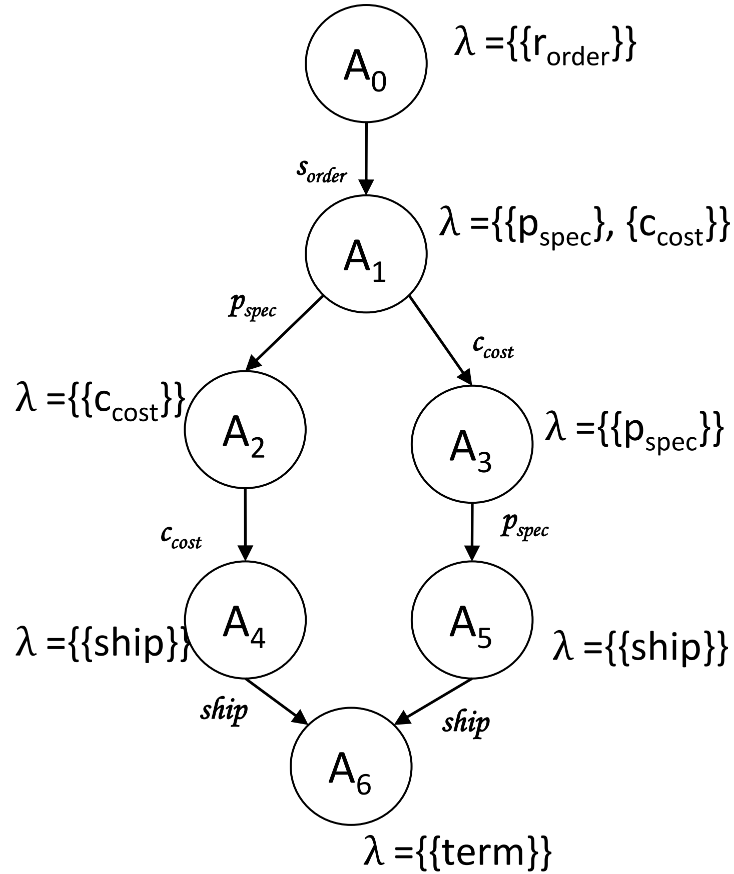


Theorem :
The composition of two SOGs $(\mathcal{G}_1, Obs_1)$ and $(\mathcal{G}_2, Obs_2)$ is a SOG $(\mathcal{G}, Obs_1 \cup Obs_2)$

Composition



SOG of contractor

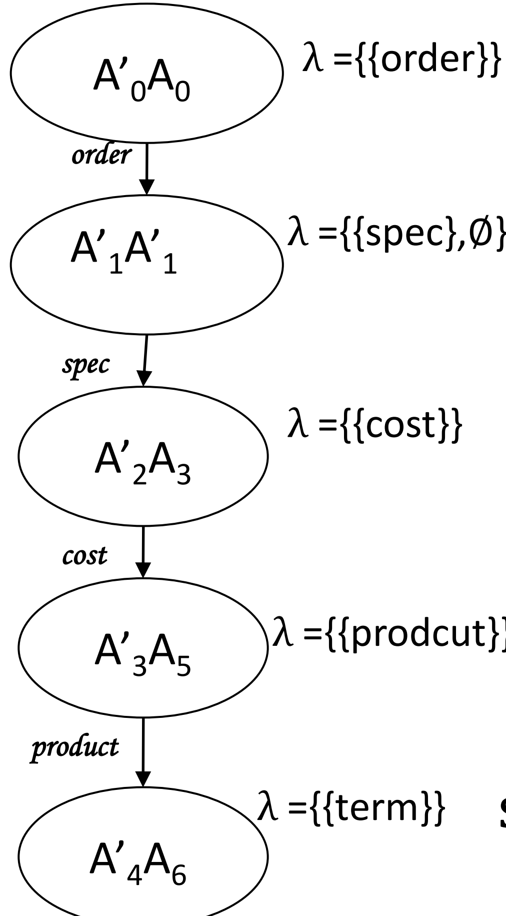


SOG of subcontractor

Composition

Reachability graph : 99 nodes + 320 edges

>

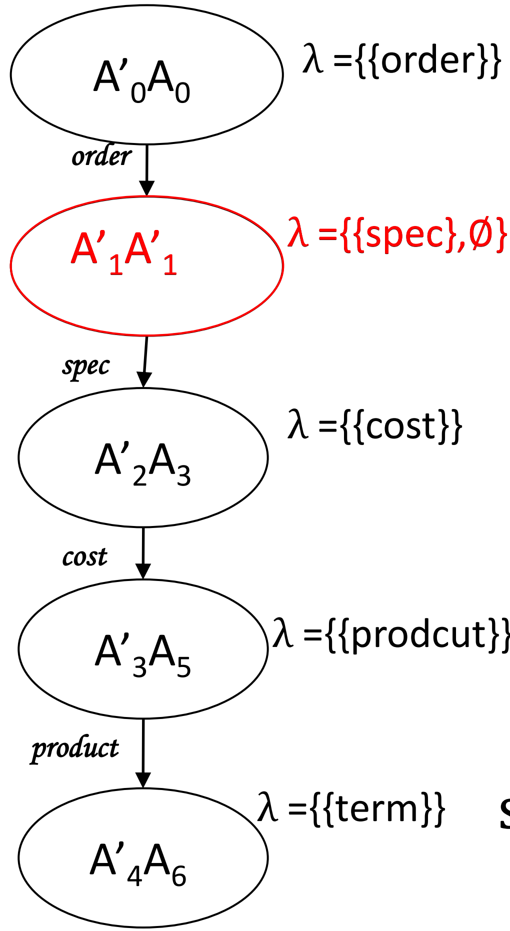


Synchronized product

Composition

Reachability graph :
99 nodes + 320 edges

>



Synchronized product

Application on web services

✓Web service : $<(P, T, F,W), m_0, I, O, \Omega>$

- **Definition (Soundness)** : $N= <(P, T, F,W), m_0, I, O, \Omega>$ is sound if :
  - ✓*option to complete:* $\forall m \in R(N^*, m_0), \exists\, m_f \in \Omega$ s.t. $m_f \in R(N^*, m_0)$
  - ✓*proper completion*: if $\exists\, m \in R(N^*, m_0)$ and $m_f \in \Omega$ s.t. $m > m_f$ then $m = m_f$;
  - ✓*no dead transitions*: $\forall\, t \in T, \exists\, m \in R(N^*, m_0)$ s.t. $m \rightarrow^t$;

- **Soundness on SOG** : $\mathcal{G}= <\mathcal{A}, Act, \rightarrow a_0, \Omega'), m_0, I, O, \Omega>$ is sound if :
  - ✓*option to complete:* $\forall a \in \mathcal{A}, \emptyset \notin a.\lambda \wedge \exists\, a_f \in \Omega'$ s.t. $a_f \in R(a)$
  - ✓*proper completion*: if $\exists\, a \in \mathcal{A}, m \in a.S, m_f \in \Omega'$ s.t. $m > m_f$ then $m = m_f$;
  - ✓*no dead transitions*: $\bigcup_{a \in \mathcal{A}} Enable(a.S):T$;

Application on web services

- **Checking Soundness on the composition of SOGs** :

Let N_1 and N_2 be two oWF-nets locally sound and let $\mathcal{G}_1$ and $\mathcal{G}_2$ be the corresponding SOGs respectively.

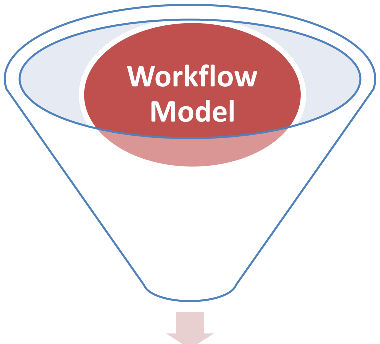
$N_1 \oplus N_2$ is sound **iff :**

✓ none $\nexists$a aggregate in $\mathcal{G}_1 \oplus \mathcal{G}_2$ s.t $\emptyset \in a.\lambda$

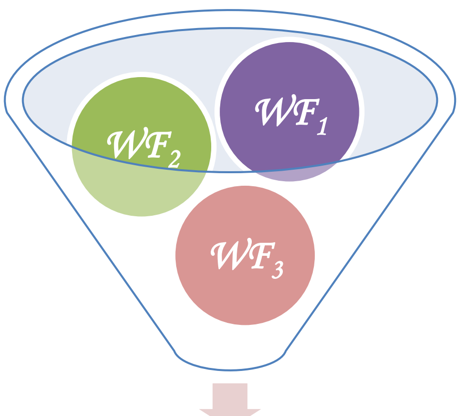
AND

✓ $\forall t \in Obs_1 \cup Obs_2$, $\exists$a, a' two aggregates in $\mathcal{G}_1 \oplus \mathcal{G}_2$ s.t. $a \rightarrow_t a'$.

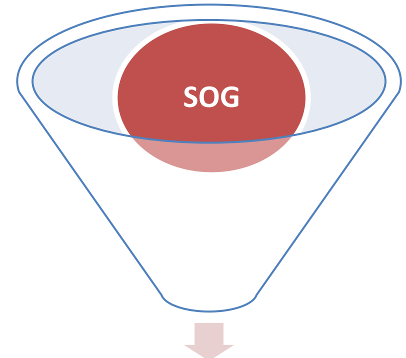
Implementation



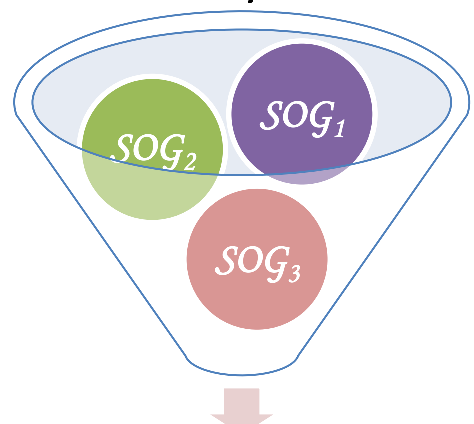
Assoiated SOG + Verification on the fly



Synchronized product of SOGs + Verification on the fly



Verification on the fly



Synchronized product of SOGs + Verification on the fly

Experimental results

Model	Places	Trans	Obs	RG		OG			SOG		
				States	Edges	States	Edges	Time(s)	States	Edges	Time(s)
C	18	11	4	26	66	12	20	<1	5	4	<1
SC	15	9	4	11	11	9	11	<1	7	7	<1
OS	15	8	8	10	10	12	17	<1	10	10	<1
R	38	33	17	28	33	369	14 E^2	<1	17	17	<1
Ph5	36	16	10	417	10 E^2	14 E^2	34 E^2	16	297	721	8
Ph6	43	19	12	14 E^2	46 E^2	61 E^2	17 E^3	245	991	28 E^2	42
Ph7	50	22	14	52 E^2	19 E^3	26 E^2	88 E^3	42 E^2	33 E^2	11 E^3	162
Ph10	71	31	20	23 E^5	23 E^4	-	-	-	12 E^4	58 E^4	15 E^2
2xPh5	71	31	4	23 E^5	23 E^4	-	-	-	21	50	15

Table: Experimental results: OG vs. SOG

-*RG:* Reachability Graph -*OG:* Operating Guideline -*SOG*: Symbolic Observation Graph

Experimental results

Model	Places	Trans	Obs	RG		OG			SOG		
				States	Edges	States	Edges	Time(s)	States	Edges	Time(s)
C	18	11	4	26	66	12	20	<1	5	4	<1
SC	15	9	4	11	11	9	11	<1	7	7	<1
OS	15	8	8	10	10	12	17	<1	10	10	<1
R	38	33	17	28	33	369	$14\ E^2$	<1	17	17	<1
Ph5	36	16	10	417	$10\ E^2$	$14\ E^2$	$34\ E^2$	16	297	721	8
Ph6	43	19	12	$14\ E^2$	$46\ E^2$	$61\ E^2$	$17\ E^3$	245	991	$28\ E^2$	42
Ph7	50	22	14	$52\ E^2$	$19\ E^3$	$26\ E^2$	$88\ E^3$	$42\ E^2$	$33\ E^2$	$11\ E^3$	162
Ph10	71	31	20	$23\ E^5$	$23\ E^4$	-	-	-	$12\ E^4$	$58\ E^4$	$15\ E^2$
2xPh5	71	31	4	$23\ E^5$	$23\ E^4$	-	-	-	21	50	15

Table: Experimental results: OG vs. SOG

-*RG:* Reachability Graph -*OG:* Operating Guideline -*SOG*: Symbolic Observation Graph

Experimental results

Model	Places	Trans	Obs	RG		OG			SOG		
				States	Edges	States	Edges	Time(s)	States	Edges	Time(s)
C	18	11	4	26	66	12	20	<1	5	4	<1
SC	15	9	4	11	11	9	11	<1	7	7	<1
OS	15	8	8	10	10	12	17	<1	10	10	<1
R	38	33	17	28	33	369	$14\ E^2$	<1	17	17	<1
Ph5	36	16	10	417	$10\ E^2$	$14\ E^2$	$34\ E^2$	16	297	721	8
Ph6	43	19	12	$14\ E^2$	$46\ E^2$	$61\ E^2$	$17\ E^3$	245	991	$28\ E^2$	42
Ph7	50	22	14	$52\ E^2$	$19\ E^3$	$26\ E^2$	$88\ E^3$	$42\ E^2$	$33\ E^2$	$11\ E^3$	162
Ph10	71	31	20	$23\ E^5$	$23\ E^4$	-	-	-	$12\ E^4$	$58\ E^4$	$15\ E^2$
2xPh5	71	31	4	$23\ E^5$	$23\ E^4$	-	-	-	21	50	15

Table: Experimental results: OG vs. SOG

-*RG:* Reachability Graph -*OG:* Operating Guideline -*SOG*: Symbolic Observation Graph

Conclusion

-Study of some approaches for abstraction workflows

-New version of the graph of symbolic observation adapted to workflow

-Checking for deadlock freeness

-CosyVerif :

- ✓Online shared tools integration platform.
- ✓Integration of ObsGraphTool :
    - Local Verification on workflow models
    - Modular verification for composition of workflows

Demo

Further work

- **Modeling, Abstraction and Verification of Inter-Enterprise Processes**

  - Consider different types of properties

  - Consider shared resources

  - Consider time explicitly

Further work

- **Modeling, Abstraction and Verification of Inter-Enterprise Processes**

  - Consider different types of properties
    - ✓ Specific properties : Expressed with temporal logic (LTL, CTL ..)
  - Consider shared resources
  - Consider time explicitly

Further work

- **Modeling, Abstraction and Verification of Inter-Enterprise Processes**

  - Consider different types of properties
    - ✓ Specific properties : Expressed with temporal logic (LTL, CTL ..)

  - Consider shared resources

  - Consider time explicitly
    - ✓ Model : e.g. timed Petri nets
    - ✓ Properties : e.g. TCTL

Bibliographie

[1] Massuthe, P. and Schmidt, K. (2005) : Operating Guidelines for Services In Proceedings of 12. Workshop "Algorithmen und Werkzeuge fur Petrinetze“

[2] Massuthe, P. Schmidt, K. and Reisig, W.(2005) : An Operating Guidelines Approach to the SOA , In Proceedings of "Annals of mathematics, computing and teleinformatics“

[3] Martens, A. (2003) : Usability of web services In Proceedings of the Fourth international conference on Web information systems engineering

[4] Martens, A. (2005) : Analysing web services based business processes. In Proceedings of the International Conference on Fundamental Approaches to Software Engineering (FASE'05)

[5] Haddad,S., Ilie, J-M. and Klai, K. (2004) : Design and Evaluation of a Symbolic and Abstraction-based Model Checker In Proceedings of Automated Technology for Verification and Analysis

[6] Klai, K. and Poitrenaud, D. (2008) : MC-SOG : An LTL Model Checker Based on Symbolic Observation Graphs In Proceedings of Petri Nets'08

[7] Klai, K. Tata, S. and Desel, J. (2009) Symbolic Abstraction and Deadlock Freeness Verification of Inter-Enterprise Processes. In Proceedings of the 7th International Conference On Business Process Management

[8] Klai, K. Tata, S. and Desel, J.(2011) Symbolic Abstraction and Deadlock-Freeness Verification of Inter-Enterprise Processes. Data & Knowledge Engeneering (DKE)

Thank you for your attention