Abstraction and Modular Verification of Services Using Symbolic Observation Graph (SOG)

Hanen OCHI

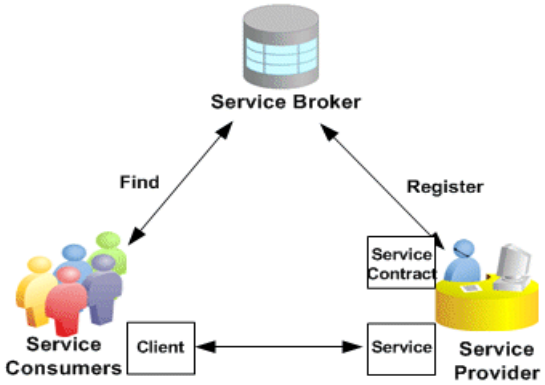
Motivation

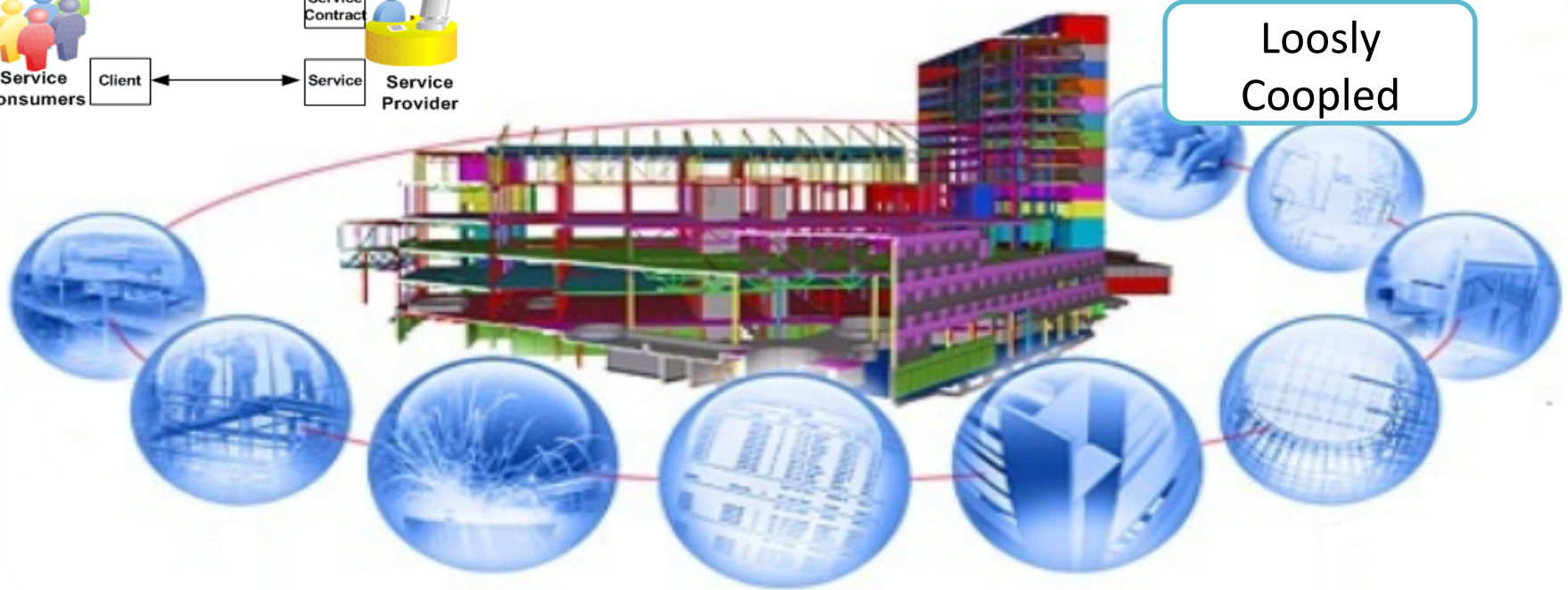
Services invoked by an organization

Workflow

Web services

Loosly Coopled

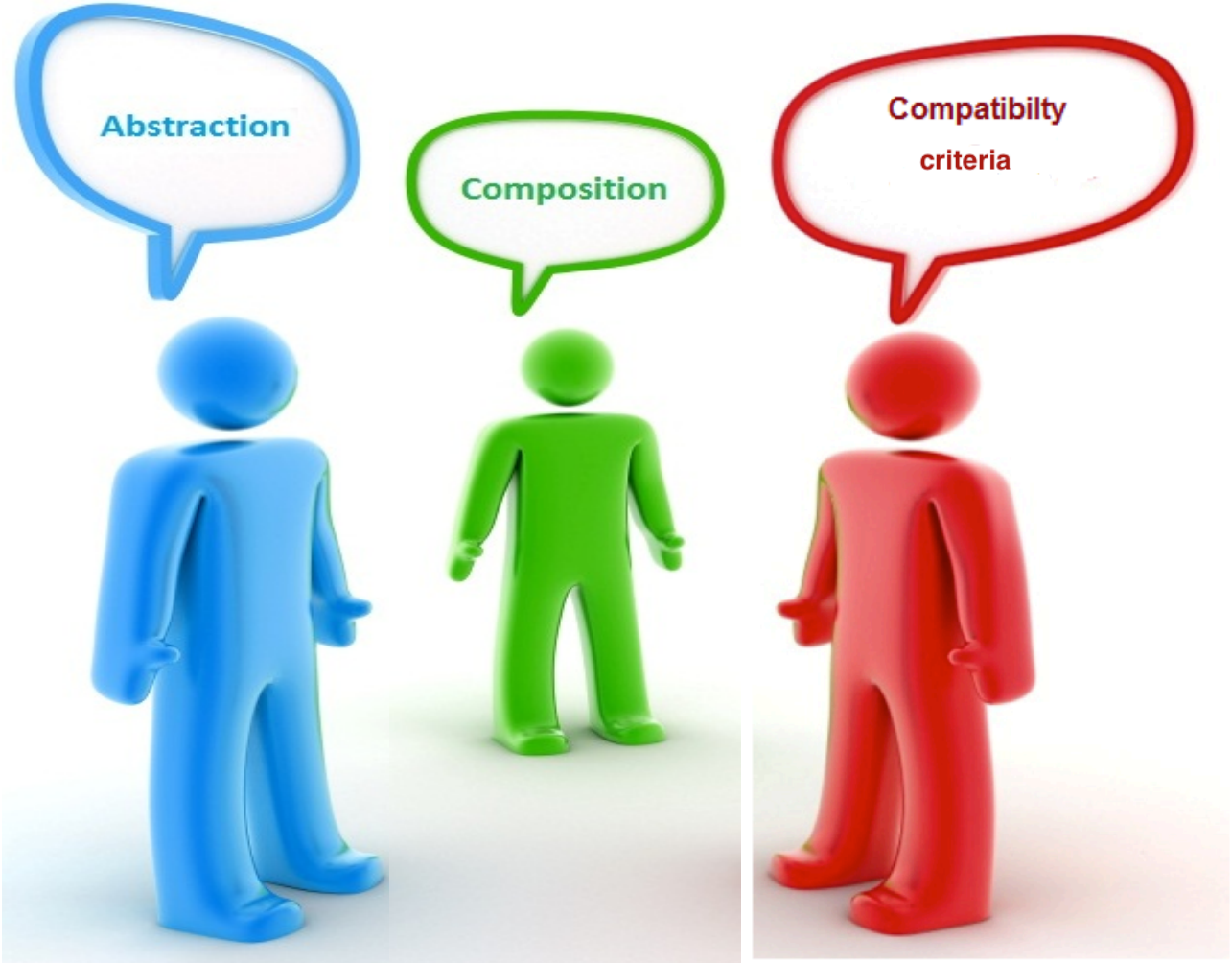




Motivation

Motivation

Motivation

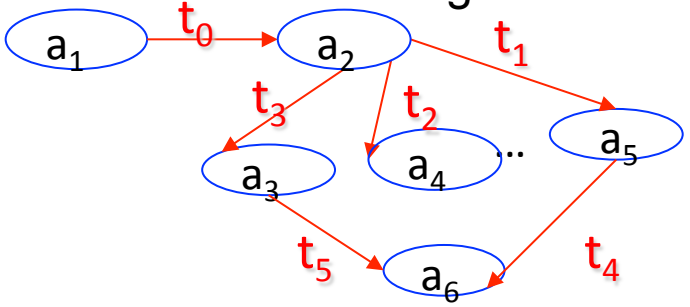


Approach based on SOG

- **Symbolic Observation Graph SOG**
  - ✓ Initially introduced for checking event-based LTL/ X properties
  - ✓Observation of only events occurring in the formula
  - **SOG:** $G(a \Rightarrow F\ b) \longrightarrow Obs=\{a, b\}$
    - ➢ Hybrid Grapn
      - ✓ Nodes (aggregates): sets of explicit states
        - Symbolic encoding (BDDs)
        - Symbolic algorithms (deadlock, livelock)
      - ✓ Edges: labelled by observed events

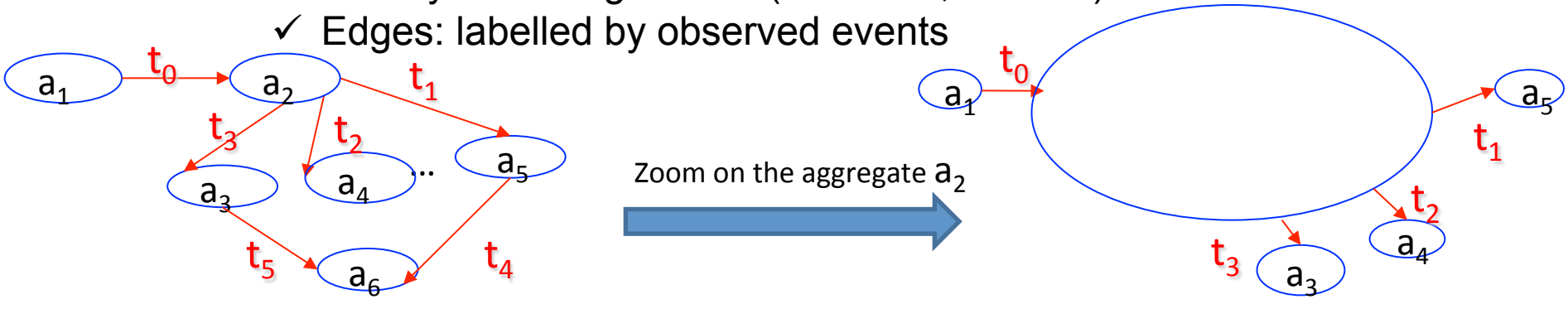
Approach based on SOG

- **Symbolic Observation Graph SOG**
  - ✓ Initially introduced for checking event-based LTL/ X properties
  - ✓ Observation of only events occurring in the formula
  - **SOG:**
    - ➢ Hybrid Graph
      - ✓ Nodes (aggregates): sets of explicit states
        - Symbolic encoding (BDDs)
        - Symbolic algorithms (deadlock, livelock)
      - ✓ Edges: labelled by observed events



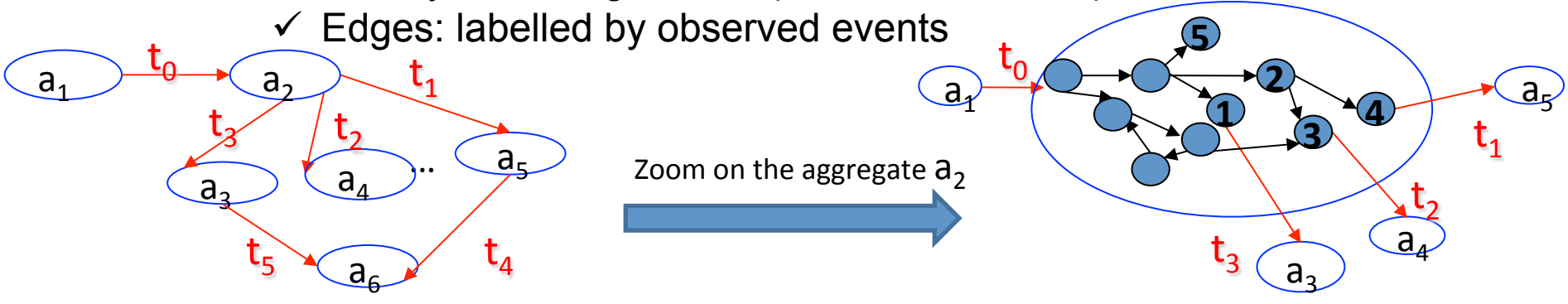
Approach based on SOG

- **Symbolic Observation Graph SOG**
  - ✓ Initially introduced for checking event-based LTL/ X properties
  - ✓ Observation of only events occurring in the formula
  - **SOG:**
    - ➢ Hybrid Graph
      - ✓ Nodes (aggregates): sets of explicit states
        - Symbolic encoding (BDDs)
        - Symbolic algorithms (deadlock, livelock)
      - ✓ Edges: labelled by observed events



Approach based on SOG

- **Symbolic Observation Graph SOG**
  - ✓ Initially introduced for checking event-based LTL/ X properties
  - ✓ Observation of only events occurring in the formula
  - **SOG:**
    - ➢ Hybrid Graph
      - ✓ Nodes (aggregates): sets of explicit states
        - Symbolic encoding (BDDs)
        - Symbolic algorithms (deadlock, livelock)
      - ✓ Edges: labelled by observed events



Approach based on SOG

- **Symbolic Observation Graph SOG**

✓ Abstraction of the behavior of service

✓Observation of only collaborative actions

Actions of the collaboration: ***Obs***

Actions of the internal behavior: ***UnObs***

Our Bottom-Up approach

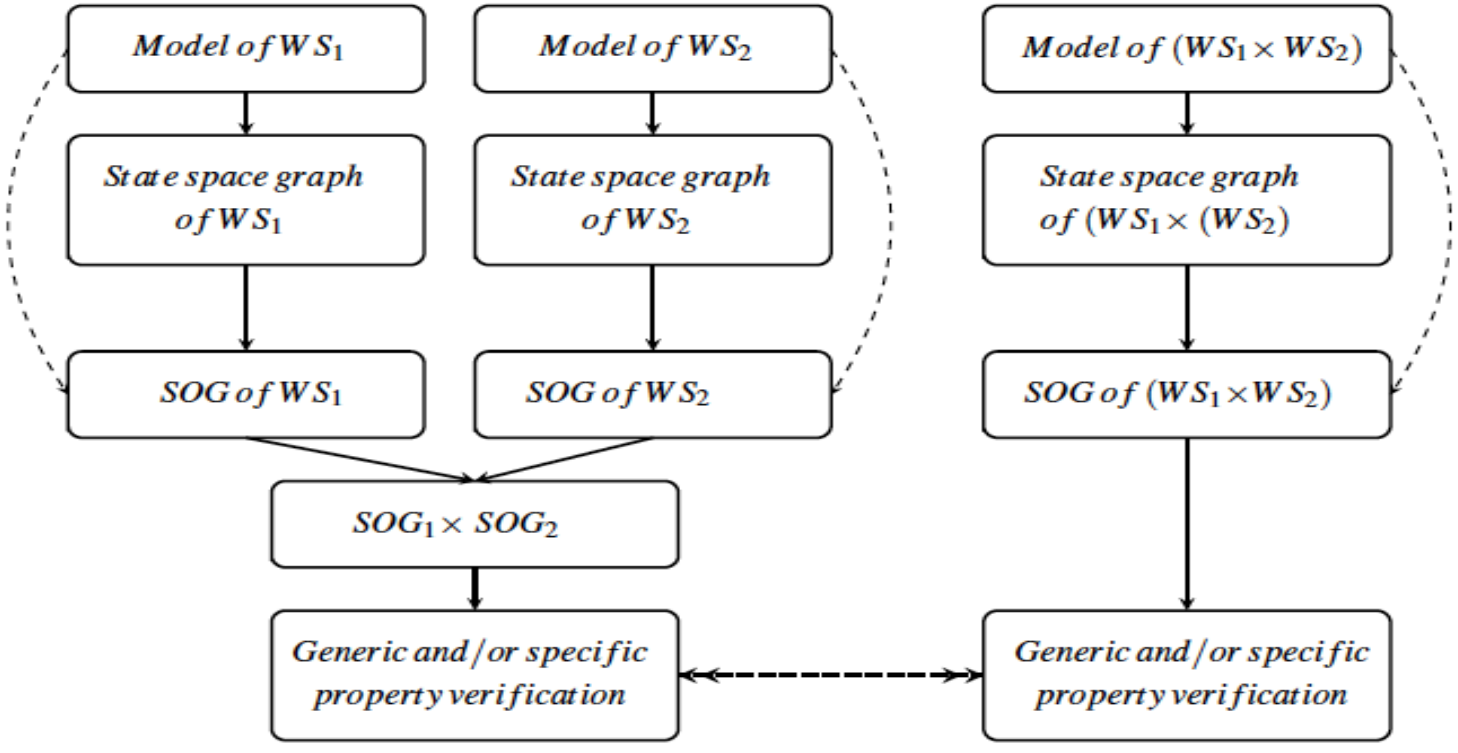
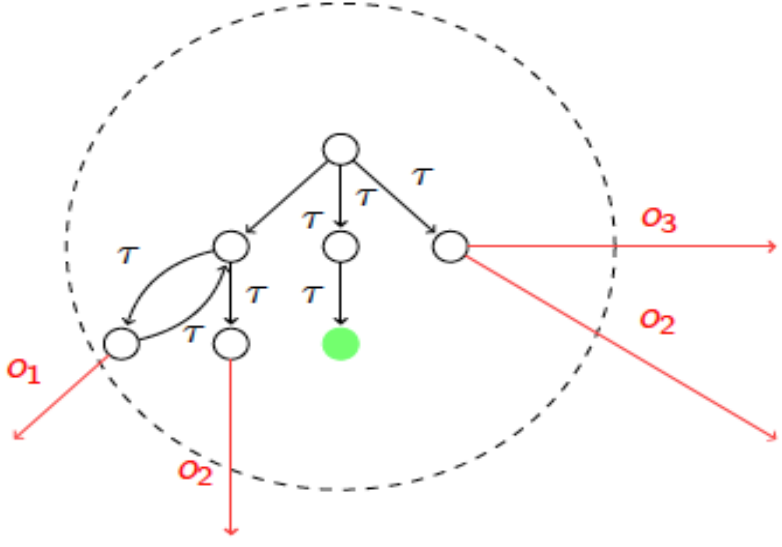
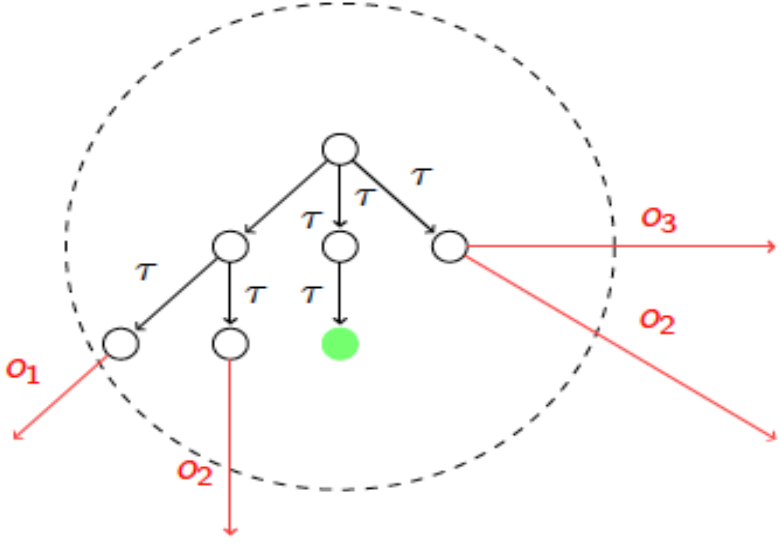


Figure 1. Schema of our approach

Approach based on SOG

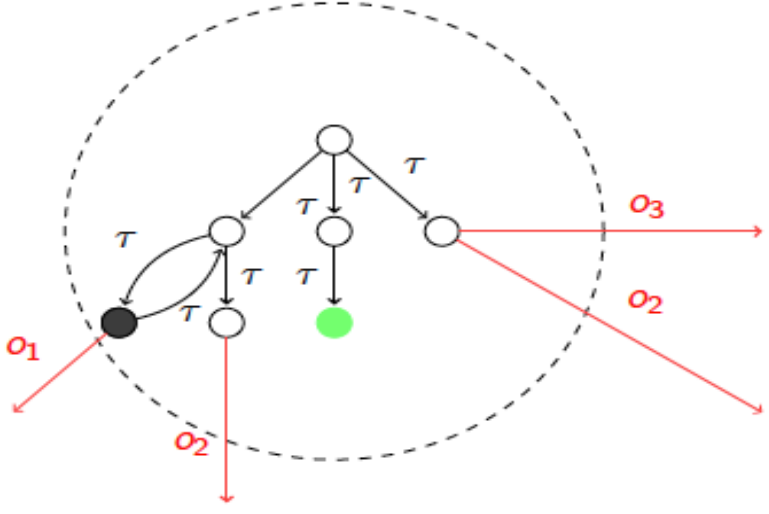
- **Composition of SOGs**

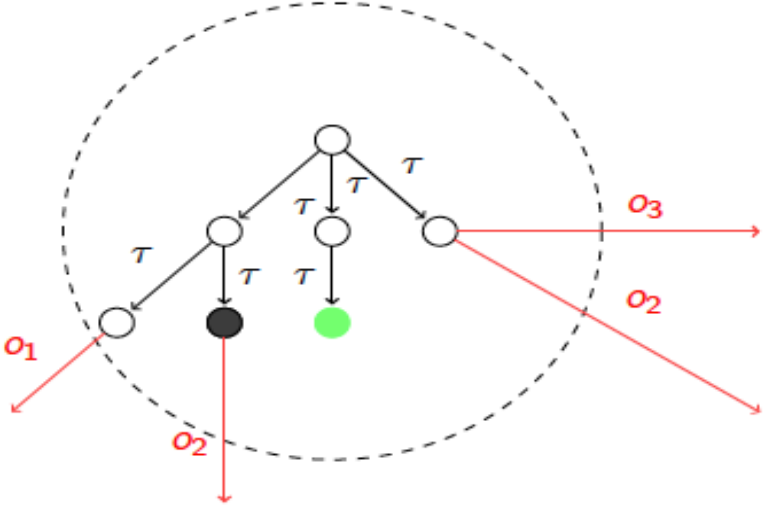




Approach based on SOG

- **Composition of SOGs**



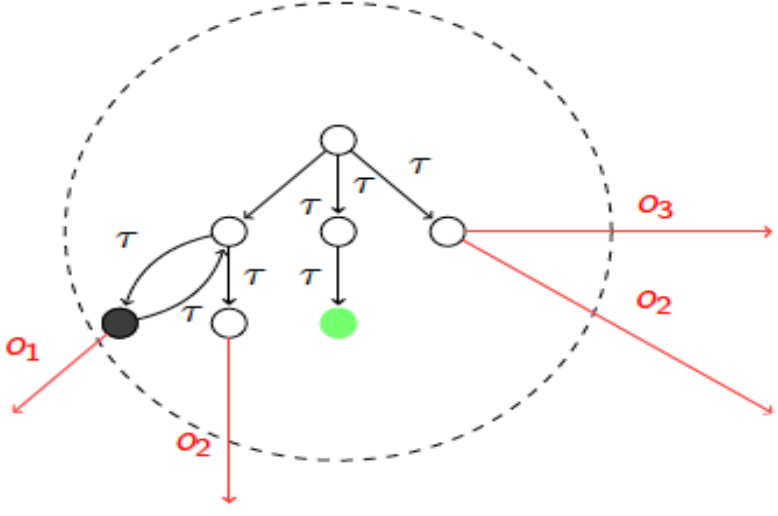


- Interlock: Deadlock caused by the interaction

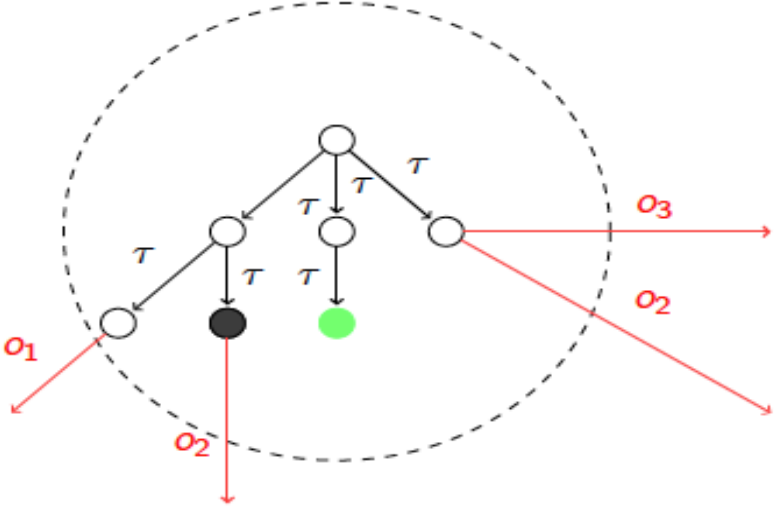
Approach based on SOG

- **Observed behavior λ**

(!) A set of sets of transitions with which we can leave the agregate



$\lambda(a_1) : \{\{o_1\}, \{o_2\}, \{term\}\}$

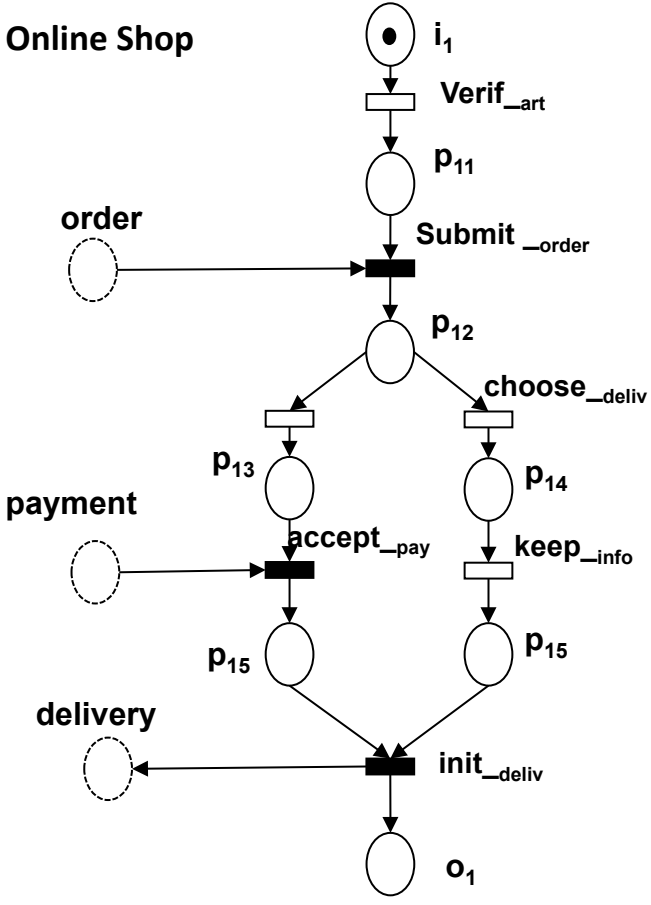


$\lambda(a_2) : \{\{o_1\}, \{o_2\}, \{term\}\}$

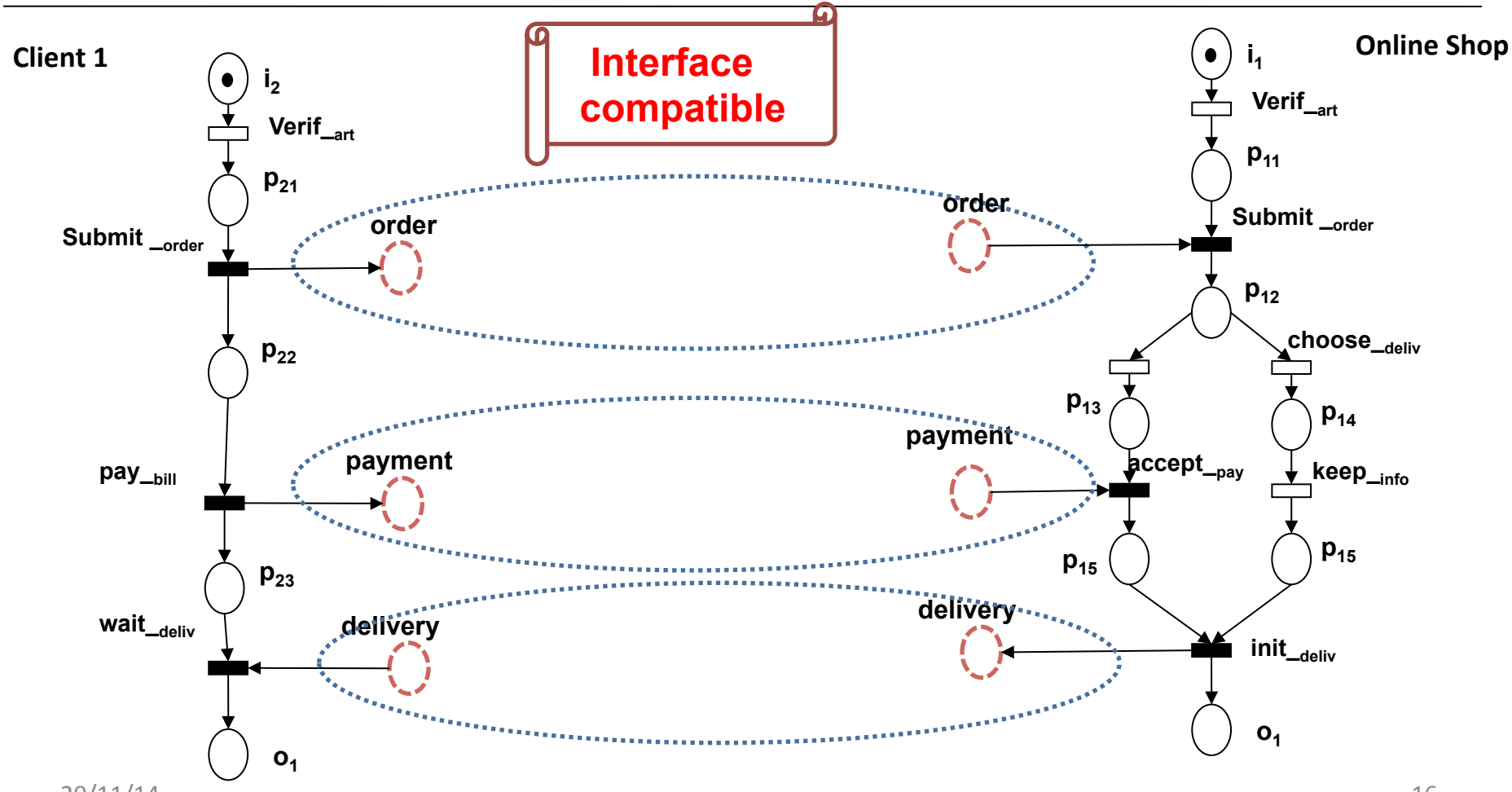
- Interlock: Deadlock caused by the interaction

Exemple of Web services

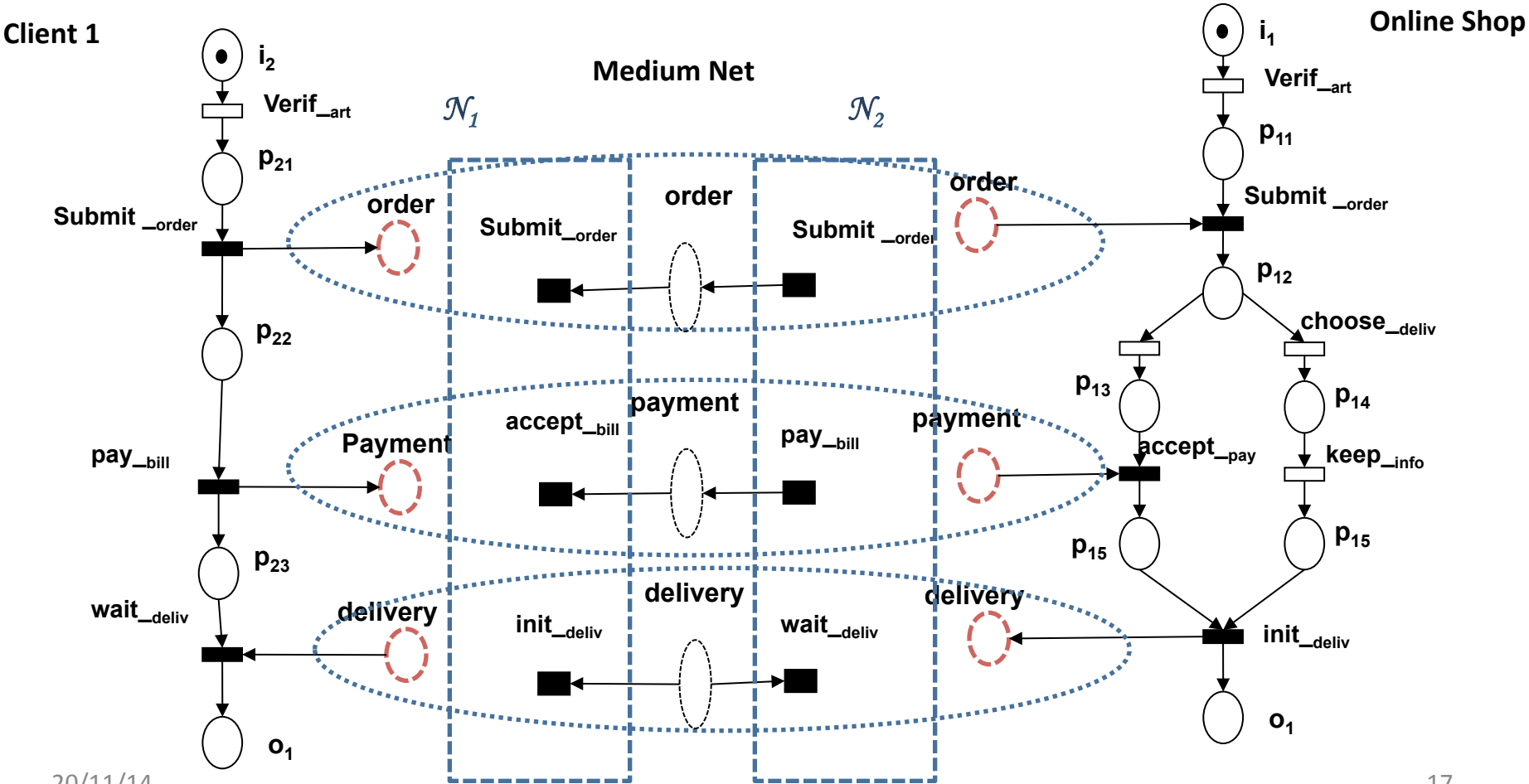
- **Formal representation :** oWF-nets
  - **Example**: Online shop



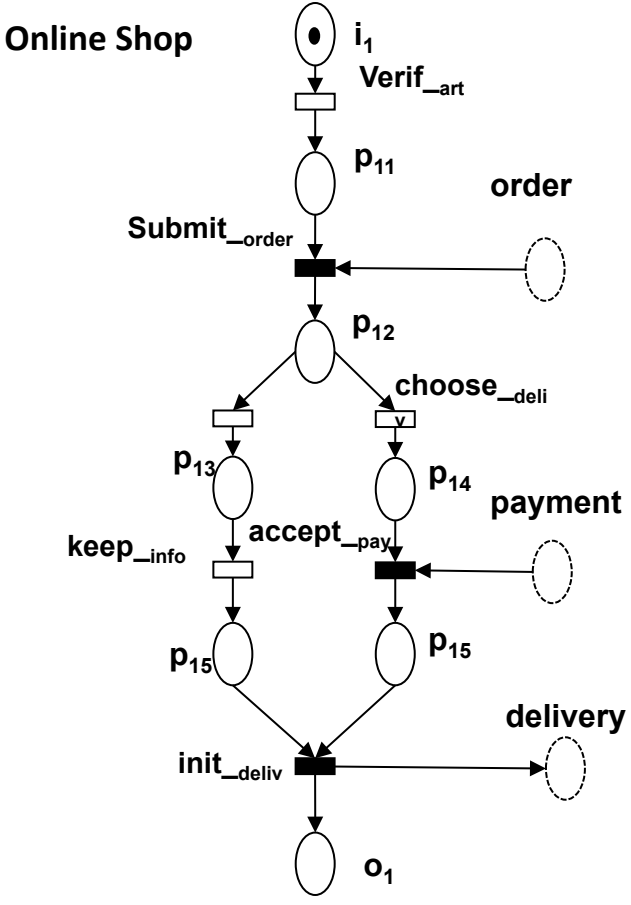
Asynchronous Composition

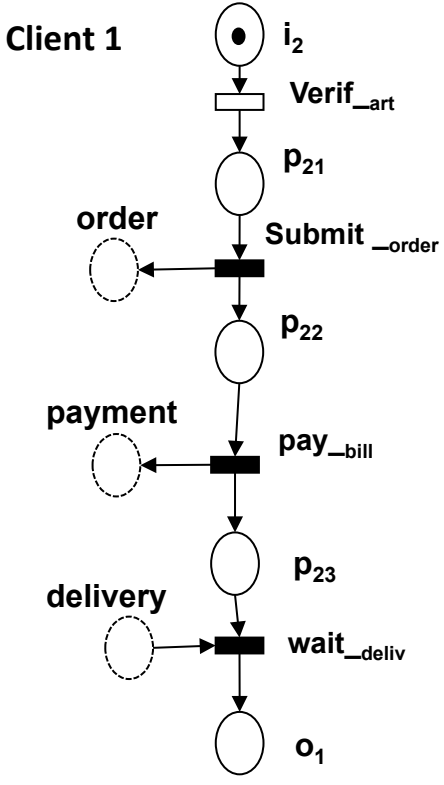


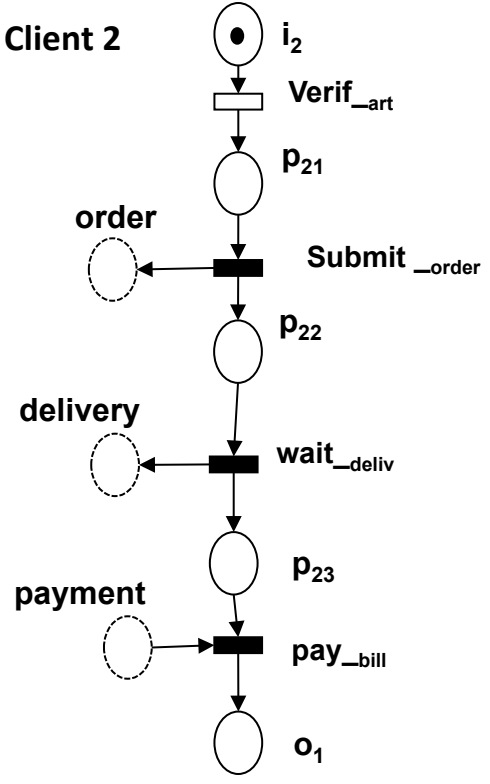
Asynchronous Composition



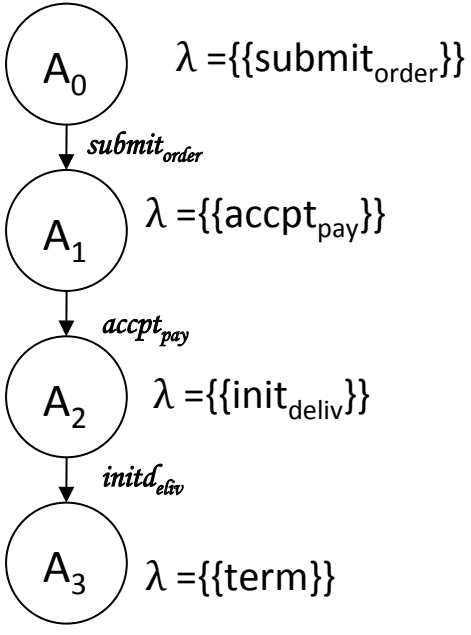
Example


Online Shop
i1
Verif_art
p11
order
Submit_order
p12
choose_deli
p13
p14
payment
keep_info
accept_pay
p15
p15
delivery
init_deliv
o1

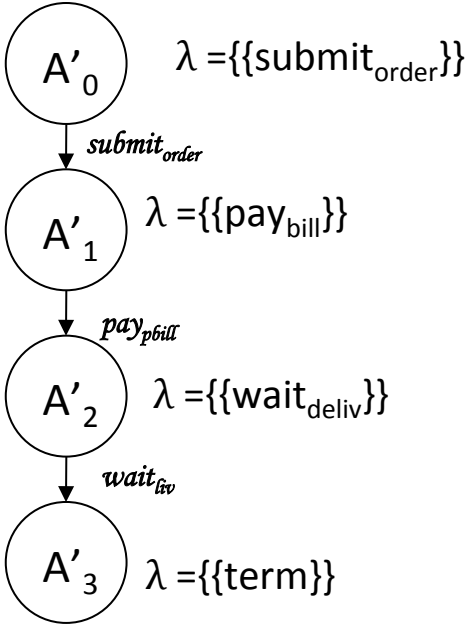

Client 1
i2
Verif_art
p21
order
Submit _order
p22
payment
pay_bill
p23
delivery
wait_deliv
o1


Client 2
i2
Verif_art
p21
order
Submit _order
p22
delivery
wait_deliv
p23
payment
pay_bill
o1

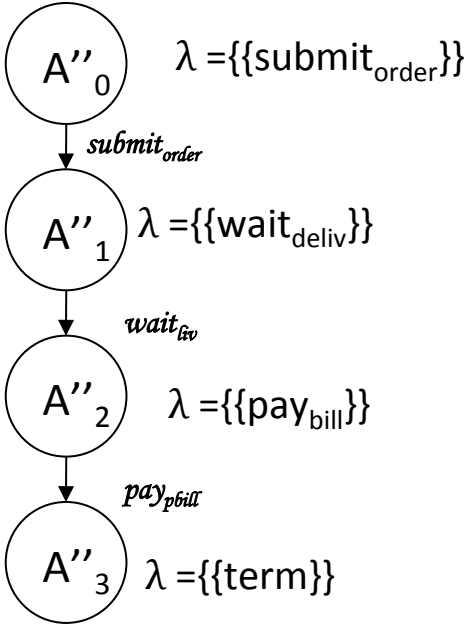
SOGs (Example)



SOG of onlineshop

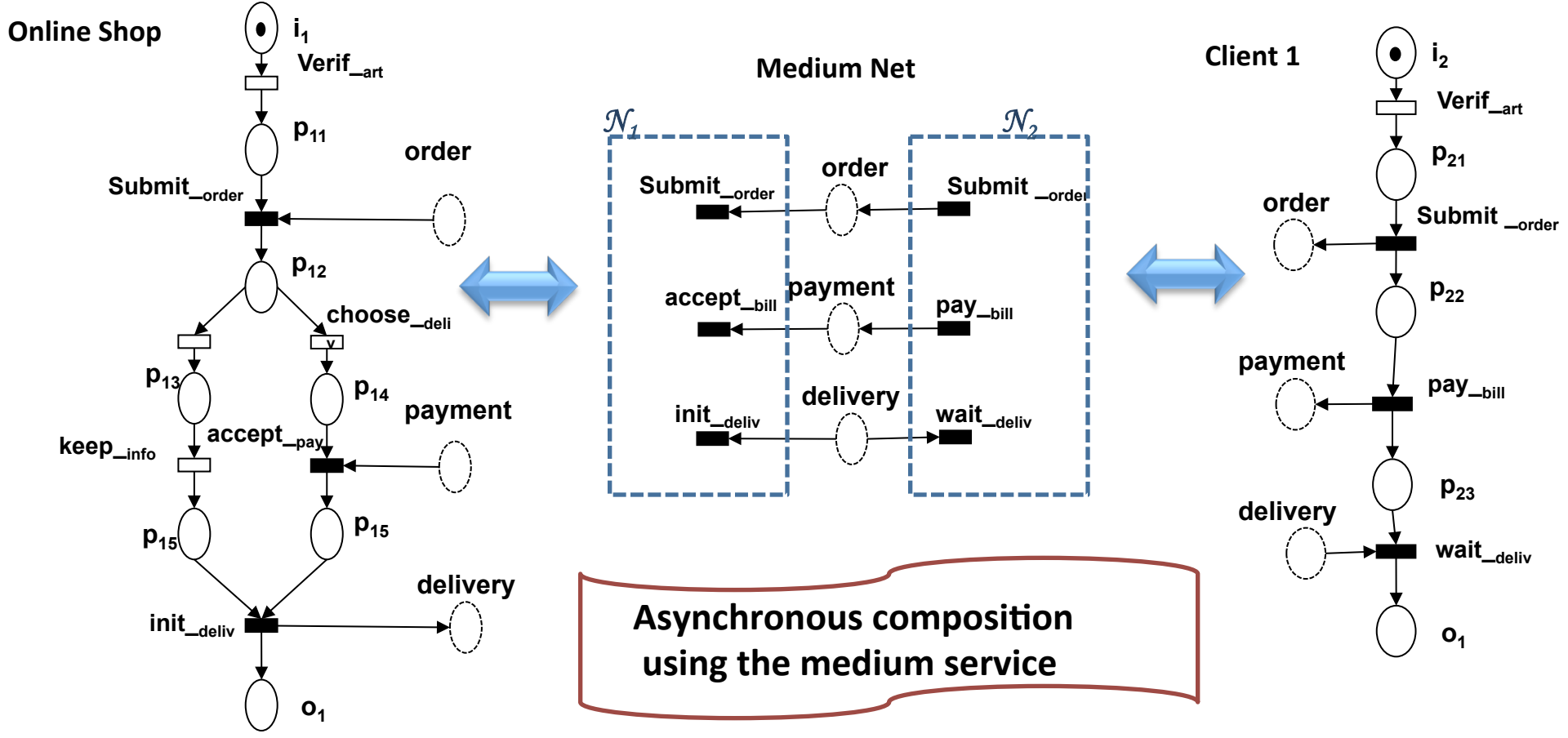


SOG of client 1



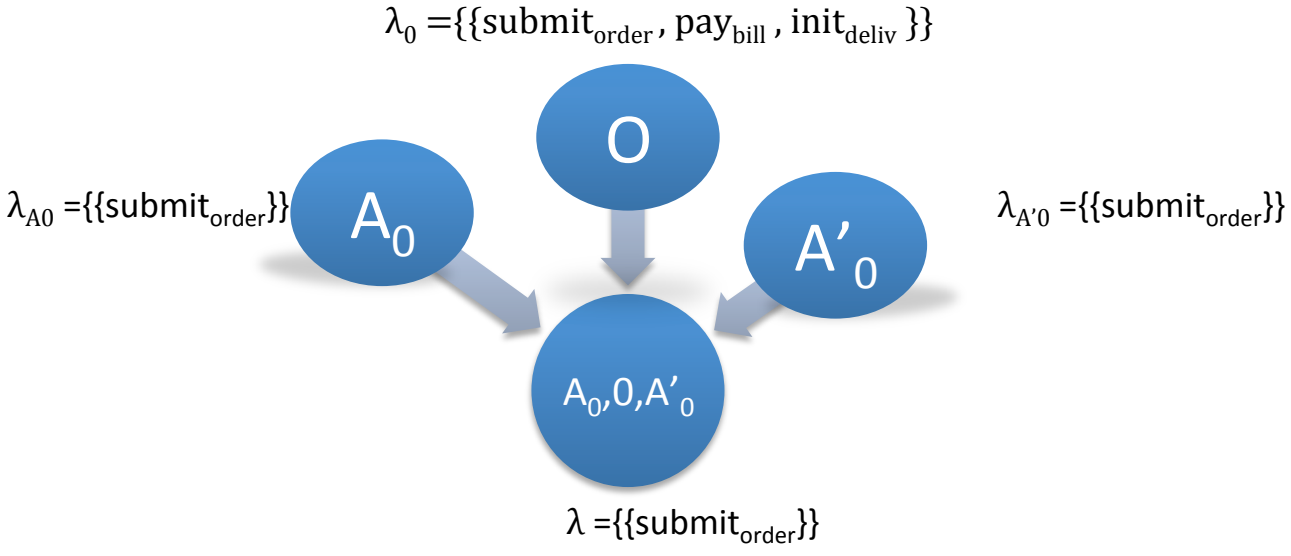
SOG of client 2

Composition



Composition of SOGs

- Asynchronous composition using the meduim service



Initial product aggregate composed by initial aggregates of the onlineshop 's SOG, the medium net's SOG and the client's SOG

Composition of SOGs

- **Theorem** :

  Let $\mathcal{WS}$ a web service and let $\mathcal{G}$ the associated SOG

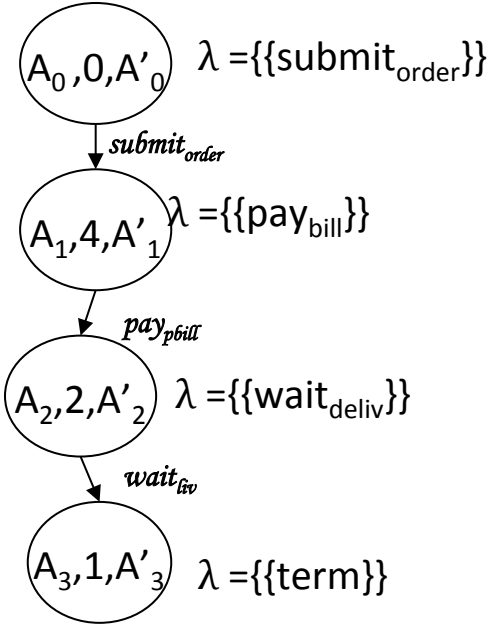
  $\mathcal{WS}$ is deadlockfree $\Leftrightarrow$ $\mathcal{G}$ is deadlockfree

Theorem :

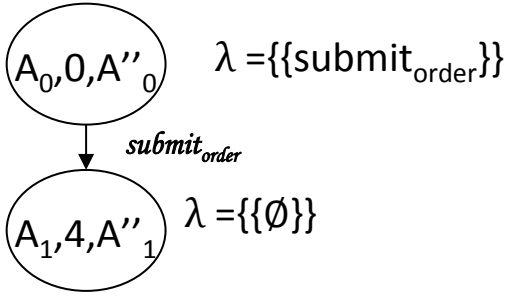
Let N_1 and N_2 be two oWF-nets and let G_1 and G_2 be the corresponding SOGs respectively :

The composition of two SOGs $(\mathcal{G}_1, Obs_1)$ and $(\mathcal{G}_2, Obs_2)$, denoted $\mathcal{G}_1 \oplus \mathcal{G}_2$, is a SOG of $\mathcal{N}_1 \bigoplus \mathcal{N}_2$

Composition

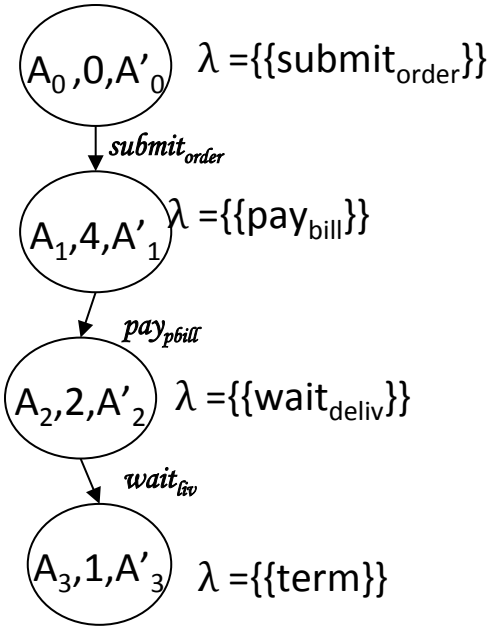


Synchronized product of SOGs (with client 1)

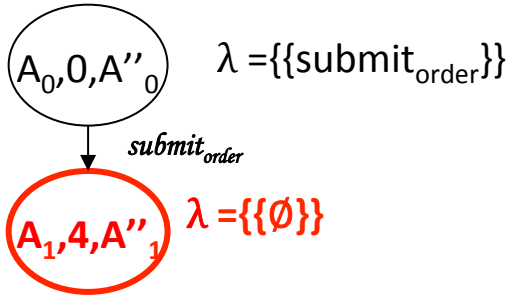


Synchronized product of SOGs (with client 2)

Checking Compatibility



Synchronized product of SOGs (with client 1)



Synchronized product of SOGs (with client 2)

- **Theorem** : Deadlock freeness

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

Some properties:

Generic properties :

Deadlockfreeness

Soundness :

- ✓ Option to complete
- ✓ Proper completion
- ✓ No dead transitions

- *Relaxed Soundness*
  - ✓each transition occur in at least one ”good” execution path.
- *Weak Soundness*
  - ✓a final marking is reachable from any reachable state
- *Easy Soundness*
  - ✓a final marking is reachable from the initial marking.

Specific properties :

Preperties expressed with LTL

Enrich aggregates with locally computed information !!!

Experimental Results

Model				Modular SOG				Non Modular SOG				LoLA		
Nom	Sound	Places	Trans	Obs	S	E	T(s)	Obs	S	E	T(s)	S	E	T(s)
C+SC1	F	25	16	4/4	9	8	<1	8	9	8	<1	22	26	<1
												1	0	<1
C+SC2	T	28	23	4/4	11	12	<1	8	11	12	<1	23	25	<1
												23	25	<1
Res	T	28	33	8/8	17	17	<1	16	17	17	<1	19	21	<1
												15	17	<1
ResTrip	F	30	26	7/5	10	12	<1	12	10	12	<1	24	27	<1
												1	0	<1

Conclusion & Further work

-Study of some approaches for abstraction web services

-New version of the graph of symbolic observation adapted to services
-And business processes

-Checking for compatibilty based on Soundness and its variant

-Implementation :
  Deadlock-freeness (integrated to CosyVerif)
  Soundness variants
  LTL modular model checking (loading ….)
- Further work :
  Consider shared resources
  Consider time explicitly

Conclusion & Further work

-Study of some approaches for abstraction web services

-New version of the graph of symbolic observation adapted to services
-And business processes

-Checking for compatibilty based on Soundness and its variant

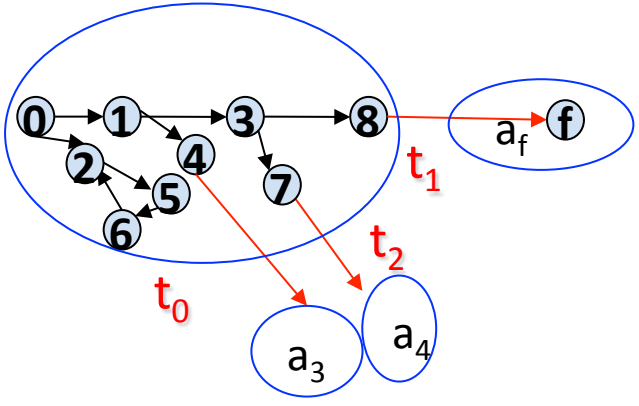
-Implementation :
  Deadlock-freeness (integrated to CosyVerif)
  Soundness variants
  LTL modular model checking (loading ….)
- Further work :
  Consider shared resources
  Consider time explicitly

Thank you for your attention

Thank you for your attention

Revisiting SOGs for modular verification of soundness properties

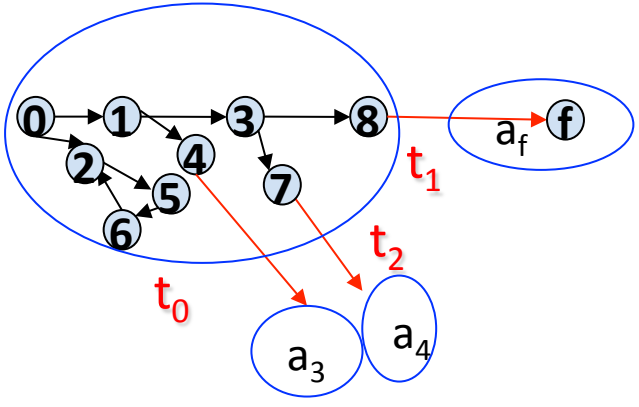
- $M_f(a) = \{m \in S \mid \exists m_f \in \Omega: m_f \in R(N,m)\}$ *is a set of marking;*
- $Enable(a.S) = \{t \in T \mid \exists m \in S: m \rightarrow^t\}$ *is a set of transitions;*



$$a = \langle S, \lambda, M_f, Enable \rangle$$

Revisiting SOGs for modular verification of soundness properties

- $M_f(a) = \{m \in S \mid \exists m_f \in \Omega\colon m_f \in R(N,m)\}$ *is a set of marking;*
- $Enable(a.S) = \{t \in T \mid \exists m \in S\colon m \rightarrow^t\}$ *is a set of transitions;*



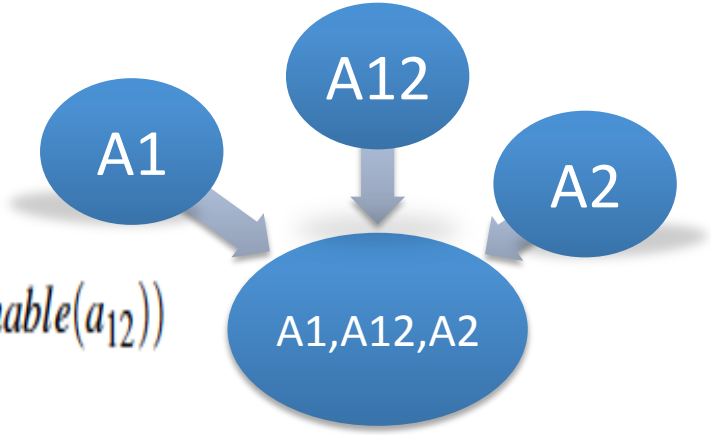
$$a = \langle S, \lambda, M_f, Enable \rangle$$

Let $G = \langle \mathcal{A}, Act, \rightarrow, a_0, \Omega' \rangle$ *be a SOG*

- ***option to complete:*** $\bigcup_{a \in \mathcal{A}} M_f(a) = \bigcup_{a \in \mathcal{A}} a.S.$
- ***proper completion:*** $\forall a \in \mathcal{A},\ \forall m \in a.S,\ \forall m_f \in \Omega,\ m \geq m_f \implies m = m_f;$
- ***no dead transitions :*** $\bigcup_{a \in \mathcal{A}} Enable(a.S) = T.$

Revisiting SOGs for modular verification of soundness properties

The Modular Verification :



$$Enable(a) = \bigcup_{i=1\ldots2}(Enable(a_i) \setminus (Obs_i \cap Obs_{12})) \cup (Enable(a_i) \cap Enable(a_{12}))$$

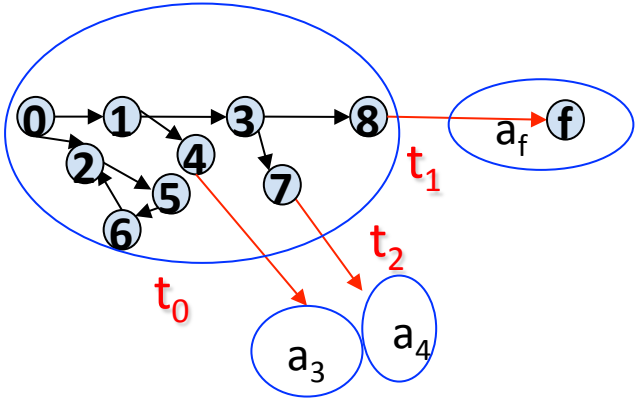
•**Theorem** :

$\mathcal{G}_i = \langle \mathcal{A}_i, a_{0i}, \rightarrow_i, \mathcal{F}_i \rangle$ $(i = 1,2)$ *be two SOGs corresponding to* $\mathcal{N}_1$ *and* $\mathcal{N}_2$

- *if* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are sound then* $\mathcal{G}_1 \oplus \mathcal{G}_2$ *is sound if the following requirements are satisfied:*
  - *option to complete:* $\forall a \in \mathcal{A}, \emptyset \notin a.\lambda \wedge \exists a_f \in \mathcal{F} \mid a_f \in R(a)$.
  - *no dead transitions :* $\bigcup_{a \in \mathcal{A}} Enable(a.S) = \bigcup_{i=1\ldots2} \bigcup_{a_i \in \mathcal{A}_i} Enable(a_i.S)$.

Revisiting SOGs for modular verification of relaxed soundness properties

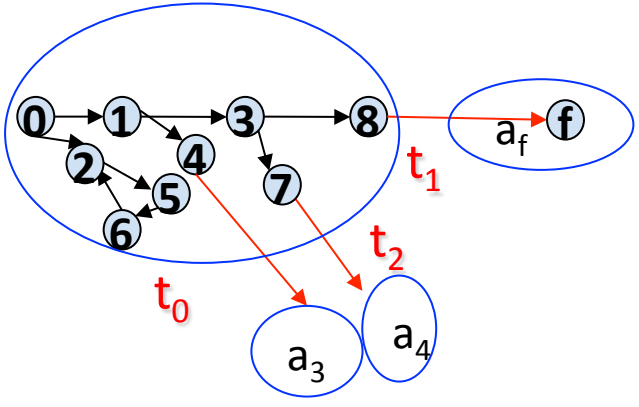
- $T_f(a) = \{t \in T \mid Succ(M_f(a), t) \cap M_f(a) \neq \emptyset\}$ *a set of transitions* $\mid Succ(S,t) = \{s' \mid \exists s \in S : s \xrightarrow{t} s'\}$ *the set of states reachable from any state of S by the firing of t.*



$$a = \langle S, \lambda, M_f, T_f \rangle$$

Revisiting SOGs for modular verification of relaxed soundness properties

- $T_f(a) = \{t \in T \mid Succ(M_f(a), t) \cap M_f(a) \neq \emptyset\}$ *a set of transitions* $\mid Succ(S,t) = \{s' \mid \exists s \in S : s \xrightarrow{t} s'\}$ *the set of states reachable from any state of S by the firing of t.*



$$a = \langle S, \lambda, M_f, T_f \rangle$$

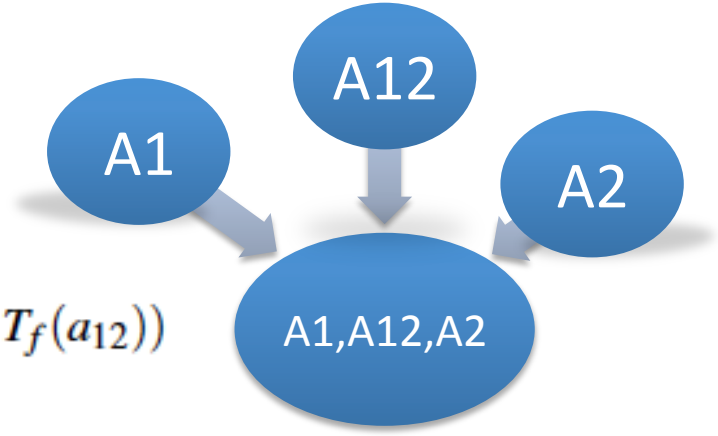
- **Theorem** :

Let $\mathcal{G} = \langle \mathcal{A}, Act, \rightarrow, a_0, \Omega' \rangle$ *be a SOG associated with an oWF-net N*

- *N is relaxed sound iff* $\bigcup_{a \in \mathcal{A}} T_f(a) = T$.

Revisiting SOGs for modular verification of relaxed soundness properties

The Modular Verification :



$$T_f(a) = \bigcup_{i=1\ldots2}(T_f(a_i) \setminus (Obs_i \cap Obs_{12})) \cup (T_f(a_i) \cap T_f(a_{12}))$$

- **Theorem** :

$\mathcal{G}_i = \langle \mathcal{A}_i, a_{0i}, \rightarrow_i, \mathcal{F}_i \rangle$ $(i = 1,2)$ *be two SOGs corresponding to* $\mathcal{N}_1$ *and* $\mathcal{N}_2$

- *if* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are relaxed sound and* $\mathcal{G}_1 \oplus \mathcal{G}_2$ *does not contain composed deadlocks, then* $\mathcal{G}_1 \oplus \mathcal{G}_2$ *is relaxed sound if* $\forall t \in Obs_1 \cup Obs_2\ \exists a \in \mathcal{A}\ \exists a_f \in \mathcal{F} : t \in T_f(a) \wedge a_f \in R(a)$ *(i.e.* $\bigcup_{a \in \mathcal{A}} T_f(a) = Obs_1 \cup Obs_2$*)*

Revisiting SOGs for modular verification of weak and easy soundness properties

Theorem :

Let $\mathcal{G} = \langle \mathcal{A}, Act, \rightarrow, a_0, \Omega' \rangle$ *be a SOG associated with an oWF-net* N

- N *is weak sound iff* $\bigcup_{a \in \mathcal{A}} M_f(a) = \bigcup_{a \in \mathcal{A}} a.S$
- N *is easy sound iff* $\bigcup_{a \in \mathcal{A}} M_f(a) \neq \emptyset$

Theorem :

if $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are weak sound then* $\mathcal{G}_1 \oplus \mathcal{G}_2$ *is weak sound* $\forall a \in \mathcal{A}, \emptyset \notin a.\lambda \wedge \exists a_f \in \mathcal{F} \mid a_f \in R(a)$.

$\mathcal{G}_1 \oplus \mathcal{G}_2$ *is easy sound if* $R(a_0) \cap \mathcal{F} \neq \emptyset$