

Structural Methods for Service Compatibility

Olivia Oanea
BEST 2009, Eindhoven
<http://service-technology.org>

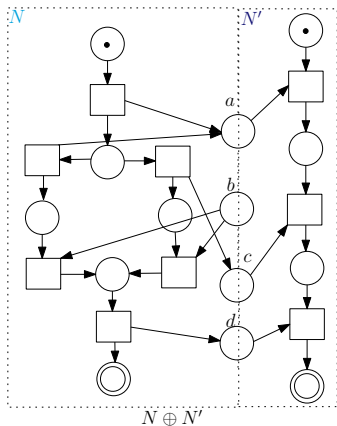
March 24, 2009



Service compatibility

deadlock-freedom

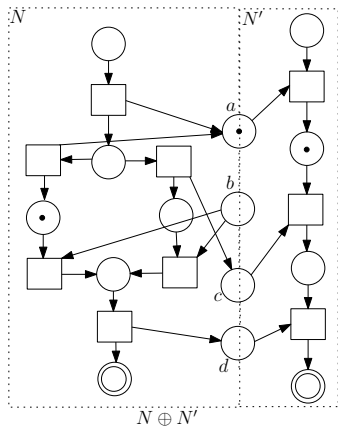
weak termination



Service compatibility

deadlock-free

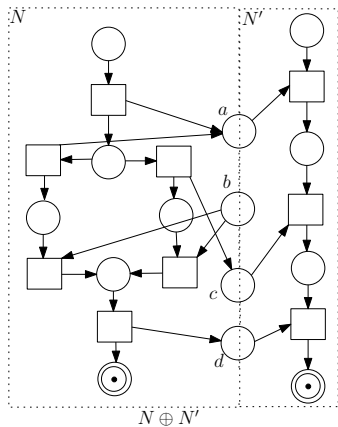
weak termination



Service compatibility

deadlock-free

weak termination

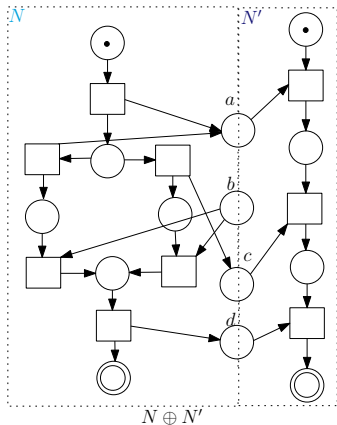


Service compatibility

deadlock-free
weak termination

brute state space analysis and diagnosis

► expensive



Service compatibility

deadlock-freedom
weak termination

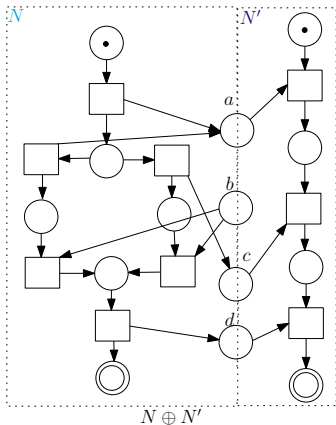
brute state space analysis and diagnosis

- ▶ expensive

compositional verification

- ▶ To what extent do individual services behave "correctly" (without interacting)?
- ▶ How can we enforce "good" communication patterns?

works in special cases



Outline

A necessary condition for weak termination

- The state equation

- Adaptability

- Weak termination with constraints

A sufficient condition for deadlock-freedom

- The siphon-trap property for service composition

- Efficient checks

Outline

A necessary condition for weak termination

- The state equation

- Adaptability

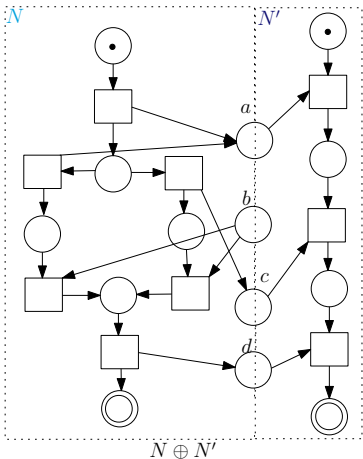
- Weak termination with constraints

A sufficient condition for deadlock-freedom

- The siphon-trap property for service composition

- Efficient checks

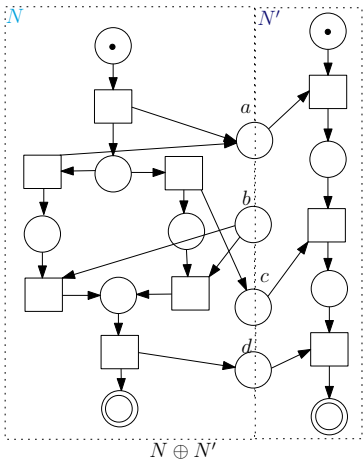
A necessary condition for weak termination



Reachability of $m_F + m'_F$ from $m_I + m'_I$ implies that the state equation \mathcal{RWS} has a solution

$$\mathcal{RWS} : \begin{pmatrix} P \\ P' \\ P_I \end{pmatrix} \begin{pmatrix} m_f \\ m'_f \\ o \end{pmatrix} = \begin{pmatrix} P \\ P' \\ P_I \end{pmatrix} \begin{pmatrix} m_o \\ m'_o \\ o \end{pmatrix} + \begin{pmatrix} P \\ P' \\ P_I \end{pmatrix} \begin{pmatrix} C_N & o \\ o & C_{N'} \\ C_I & C'_I \end{pmatrix} \cdot \begin{matrix} T \\ T' \end{matrix} \begin{pmatrix} x \\ x' \end{pmatrix}$$

A necessary condition for weak termination



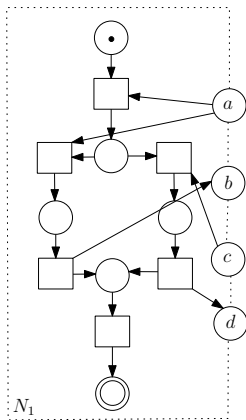
Weak termination implies $\mathcal{RW}\mathcal{S}$ has a solution

- ▶ reachability of the final marking for N and N' without interface places
- ▶ for any message, the number of sending events matches the number of receiving events

$x(b) = 1 \neq x'(b) = 0 \rightarrow$ not compatible

$$\mathcal{RW}\mathcal{S} : \begin{pmatrix} P \\ P' \\ P_i \end{pmatrix} \begin{pmatrix} m_f \\ m'_f \\ o \end{pmatrix} = \begin{pmatrix} P \\ P' \\ P_i \end{pmatrix} \begin{pmatrix} m_o \\ m'_o \\ o \end{pmatrix} + \begin{pmatrix} P \\ P' \\ P_i \end{pmatrix} \begin{pmatrix} C_N & o \\ o & C_{N'} \\ C'_1 & C'_1 \end{pmatrix} \cdot \begin{matrix} T & T' \\ T' \end{matrix} \begin{pmatrix} x \\ x' \end{pmatrix}$$

A necessary condition for adapter generation



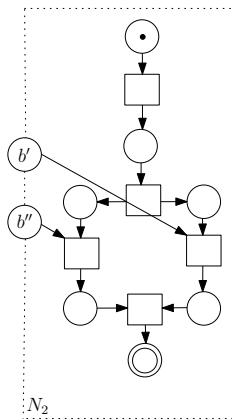
\mathcal{R}

$r_1: \mapsto a$

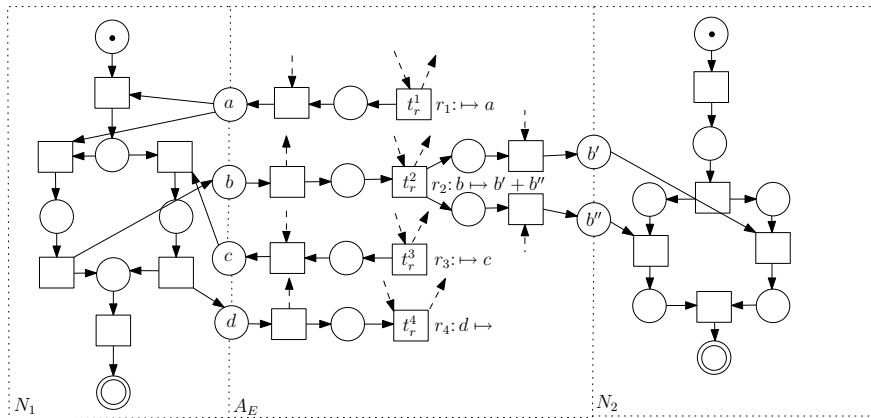
$r_2: b \mapsto b' + b''$

$r_3: \mapsto c$

$r_4: d \mapsto$



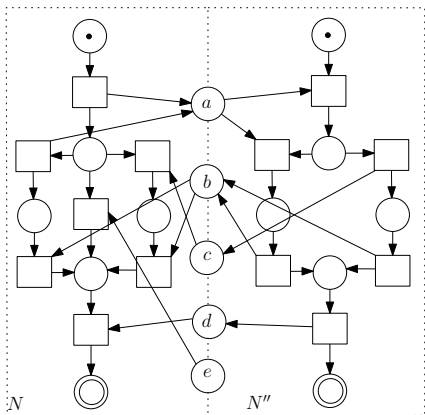
A necessary condition for adapter generation



N_1 and N_2 are adaptable by the set of rules \mathcal{R} implies

The state equation for $N_1 \oplus A_E \oplus N_2$ has a solution.

Necessary condition for compatibility with behavioral constraints



The state equation of $N \oplus N''$ and constraints for

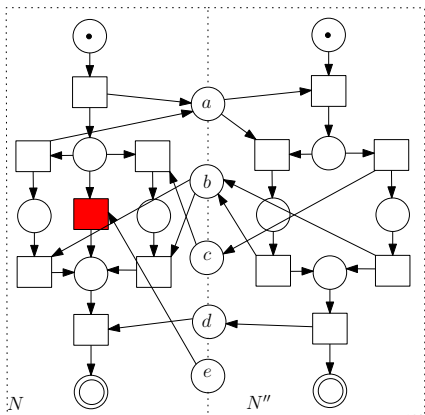
place, transition cover

event cover

free-choice sending cover

have a solution

Necessary condition for compatibility with behavioral constraints



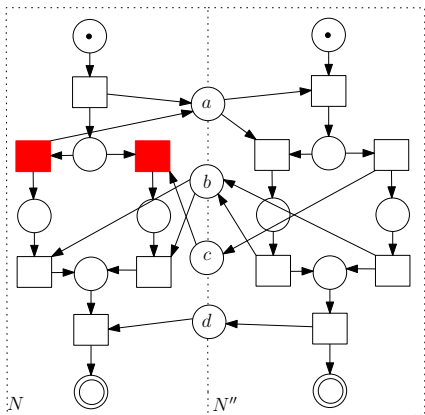
The state equation of $N \oplus N''$ and constraints for place, transition cover

event cover

free-choice sending cover

have a solution

Necessary condition for compatibility with behavioral constraints



The state equation of $N \oplus N''$ and constraints for place, transition cover

event cover

free-choice sending cover

have a solution

Outline

A necessary condition for weak termination

- The state equation

- Adaptability

- Weak termination with constraints

A sufficient condition for deadlock-freedom

- The siphon-trap property for service composition

- Efficient checks

Outline

A necessary condition for weak termination

- The state equation

- Adaptability

- Weak termination with constraints

A sufficient condition for deadlock-freedom

- The siphon-trap property for service composition

- Efficient checks

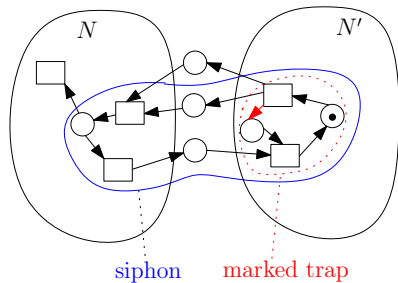
A sufficient condition for deadlock-freedom in service composition

Siphon-trap property

Each minimal siphon contains a marked trap.

Siphon-trap property for composed services

- ▶ siphons spanning over sets of interface places
- ▶ check for each siphon with at least two interface places whether it contains a marked trap



A sufficient condition for deadlock-freedom in service composition

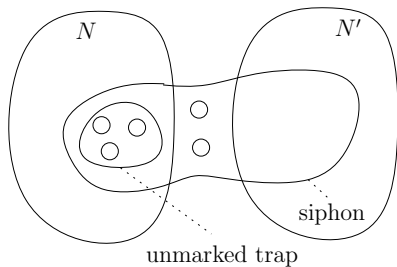
A siphon spanning over at least two interface places includes

unmarked traps (UT)

unmarked semi-traps (UST)

marked semi-traps (MST)

locally marked trap (LMT)



A sufficient condition for deadlock-freedom in service composition

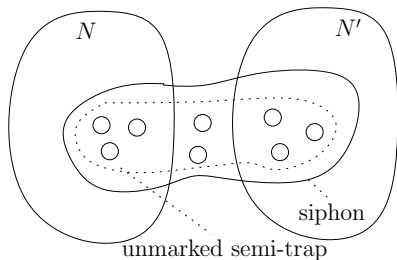
A siphon spanning over at least two interface places includes

unmarked traps (UT)

unmarked semi-traps (UST)

marked semi-traps (MST)

locally marked trap (LMT)



A sufficient condition for deadlock-freedom in service composition

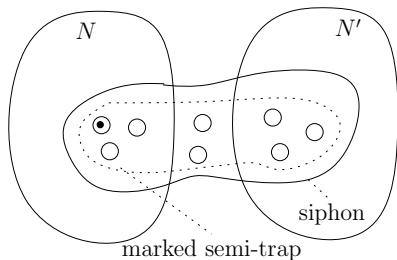
A siphon spanning over at least two interface places includes

unmarked traps (UT)

unmarked semi-traps (UST)

marked semi-traps (MST)

locally marked trap (LMT)



A sufficient condition for deadlock-freedom in service composition

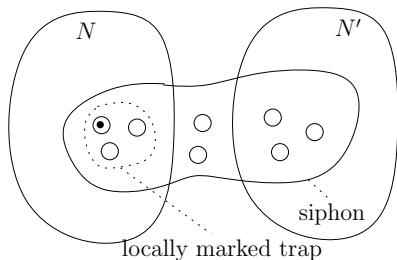
A siphon spanning over at least two interface places includes

unmarked traps (UT)

unmarked semi-traps (UST)

marked semi-traps (MST)

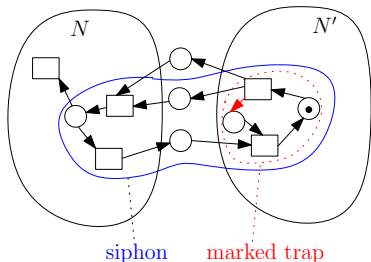
locally marked trap (LMT)



A sufficient condition for deadlock-freedom in service composition

Efficient checks

1. no UT in N implies all traps in N' are UST, LMT or MT
2. no UST and UT in N implies all traps in N' must be LMT or MST
3. no MST, UT and UST implies all traps in N' must be LMT



Conclusions

Necessary condition for weak termination of service composition using the state equation

Sufficient condition for deadlock freedom of service composition using the siphon-trap property

- ▶ Fast discovery/composition of compatible services
- ▶ Compositional analysis and diagnosis of services

Future work

- ▶ improve structural methods for compatibility
- ▶ implement the approach in current tools
- ▶ evaluation on industrial case studies

Thank you! Any questions?