Un Algoritmo per la Costruzione di Prefissi Finiti e Completi per Reti di Petri Contestuali

Alessandro Bruni

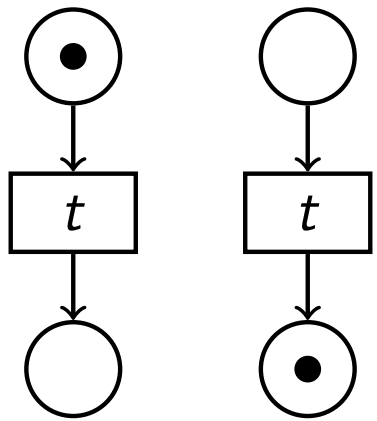
23-07-2009

Il problema

- Verifica formale di sistemi concorrenti

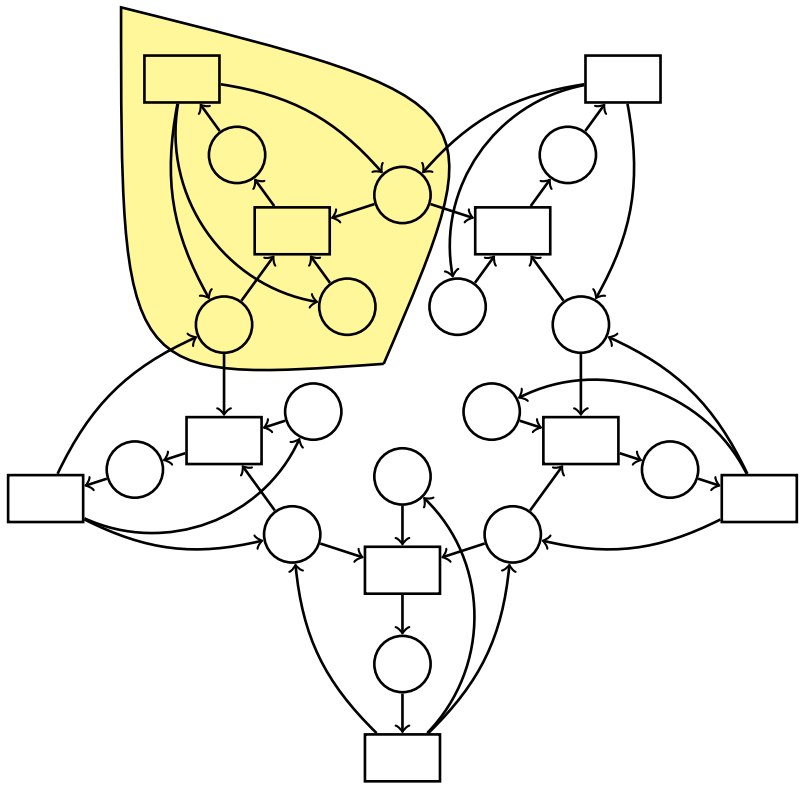
Il problema

- Verifica formale di sistemi concorrenti
- Reti di Petri: modello per la concorrenza



Il problema

- Verifica formale di sistemi concorrenti
- Reti di Petri: modello per la concorrenza



Semantiche ad ordine parziale

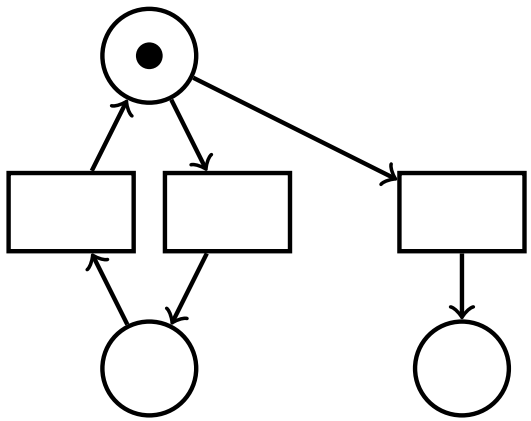
- Problema dell'esplosione dello spazio degli stati

Semantiche ad ordine parziale

- Problema dell'esplosione dello spazio degli stati
- Approccio di tipo "partial order": la concorrenza è rappresentata in modo "esplicito"

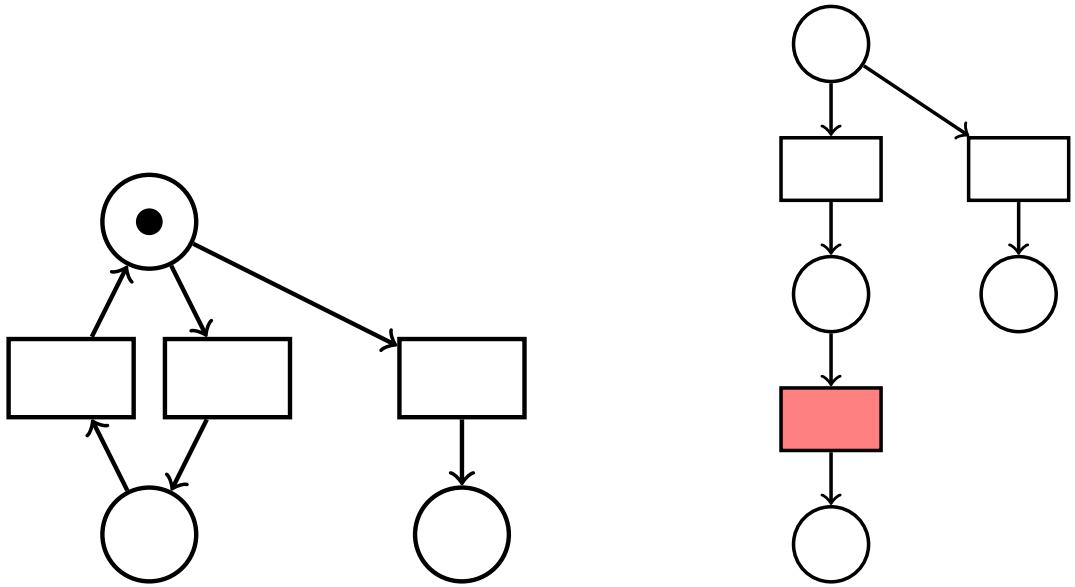
Semantiche ad ordine parziale

- Problema dell'esplosione dello spazio degli stati
- Approccio di tipo "partial order": la concorrenza è rappresentata in modo "esplicito"
- *Semantica di unfolding*: dimensione potenzialmente infinita



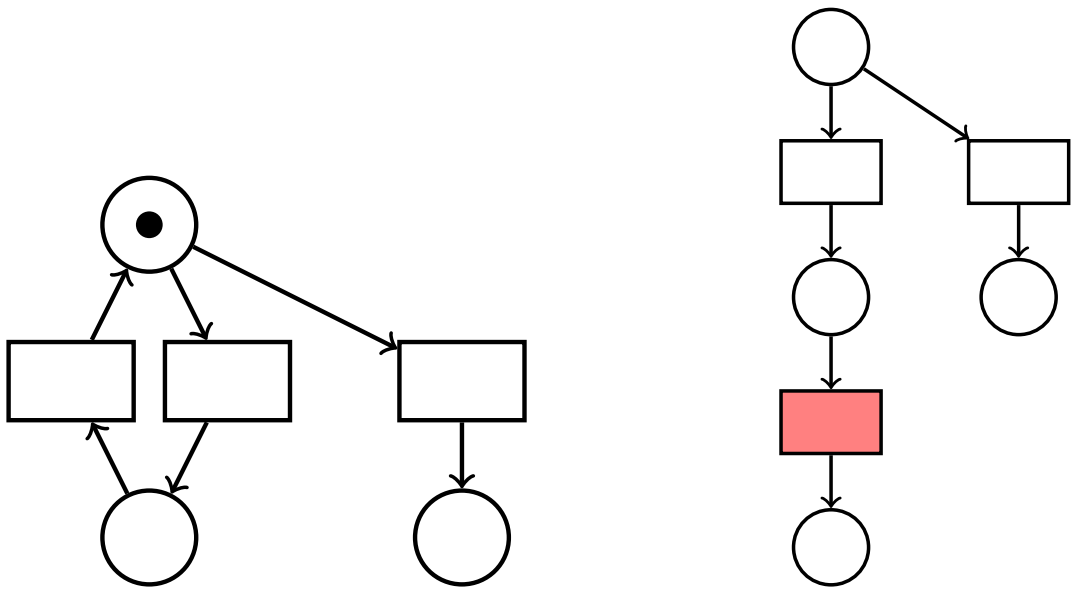
Semantiche ad ordine parziale

- Problema dell'esplosione dello spazio degli stati
- Approccio di tipo "partial order": la concorrenza è rappresentata in modo "esplicito"
- *Semantica di unfolding*: dimensione potenzialmente infinita
  1. McMillan: prefisso finito e completo



Semantiche ad ordine parziale

- Problema dell'esplosione dello spazio degli stati
- Approccio di tipo "partial order": la concorrenza è rappresentata in modo "esplicito"
- *Semantica di unfolding*: dimensione potenzialmente infinita
  1. McMillan: prefisso finito e completo
  2. Cut-off: un evento che con la sua storia causale non aggiunge informazione



Reti di Petri Contestuali

- Aggiungono il “contesto”: le transizioni verificano la presenza di token nelle piazze da leggere senza consumarli.

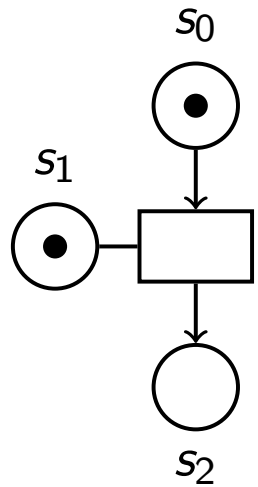


Figura: Scatto di una rete di Petri contestuale

Reti di Petri Contestuali

- Aggiungono il “contesto”: le transizioni verificano la presenza di token nelle piazze da leggere senza consumarli.

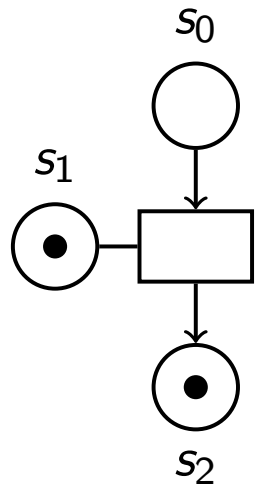


Figura: Scatto di una rete di Petri contestuale

Reti di Petri Contestuali

- Aggiungono il “contesto”: le transizioni verificano la presenza di token nelle piazze da leggere senza consumarli.
- L’algoritmo di McMillan fallisce nel caso contestuale

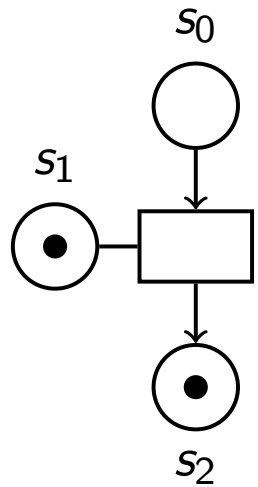


Figura: Scatto di una rete di Petri contestuale

Reti di Petri Contestuali

- Aggiungono il “contesto”: le transizioni verificano la presenza di token nelle piazze da leggere senza consumarli.
- L’algoritmo di McMillan fallisce nel caso contestuale
- Un evento ha più “storie causali” ed occorre considerarle

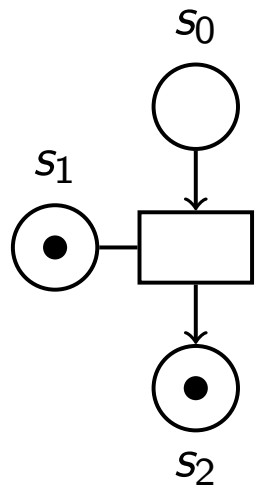


Figura: Scatto di una rete di Petri contestuale

Reti di Petri Contestuali

- Aggiungono il "contesto": le transizioni verificano la presenza di token nelle piazze da leggere senza consumarli.
- L'algoritmo di McMillan fallisce nel caso contestuale
- Un evento ha più "storie causali" ed occorre considerarle
- Algoritmo in [BCKS08] per il calcolo diretto del prefisso finito e completo di reti contestuali, senza codifica

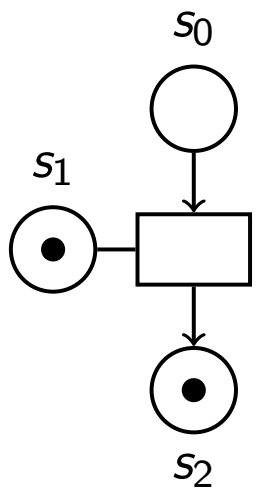
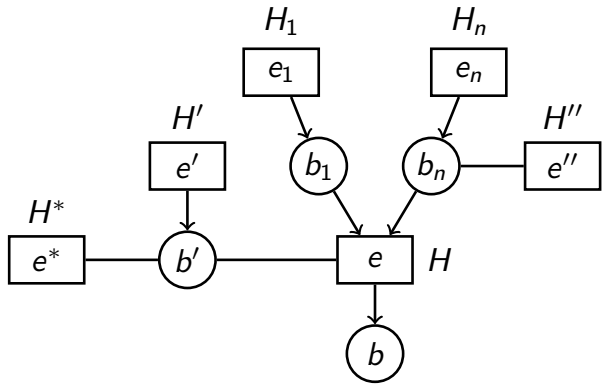


Figura: Scatto di una rete di Petri contestuale

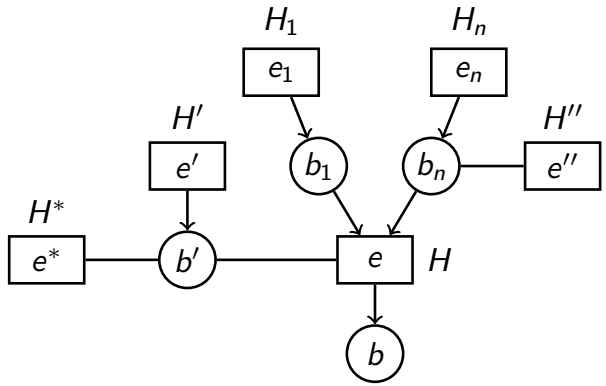
Gestione della Concorrenza

- *Idea:* storie associate alle piazze (non alle transizioni) per rappresentare la concorrenza.



Gestione della Concorrenza

- *Idea:* storie associate alle piazze (non alle transizioni) per rappresentare la concorrenza.

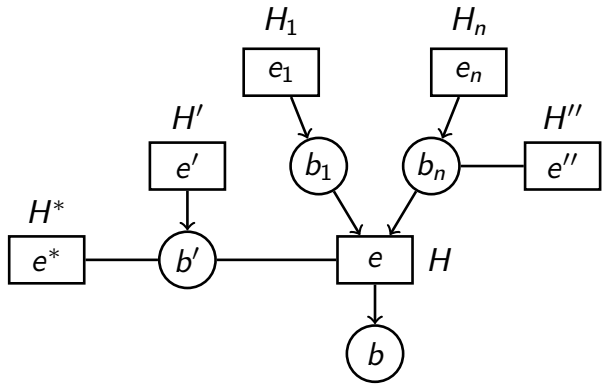


- Una storia è composta da sottostorie:

$$H_e = e \cup \bigcup_i H_i \text{ s.t. } H_i \in Hist(b), \exists b \in {}^\bullet e \cup \underline{e}$$

Gestione della Concorrenza

- *Idea:* storie associate alle piazze (non alle transizioni) per rappresentare la concorrenza.



- Una storia è composta da sottostorie:

$$H_e = e \cup \bigcup_i H_i \text{ s.t. } H_i \in Hist(b), \exists b \in {}^\bullet e \cup \underline{e}$$

concurrency $\langle b, H\rangle co \langle b', H'\rangle \iff$ H and H' are consistent and after their execution b and b' are both marked.

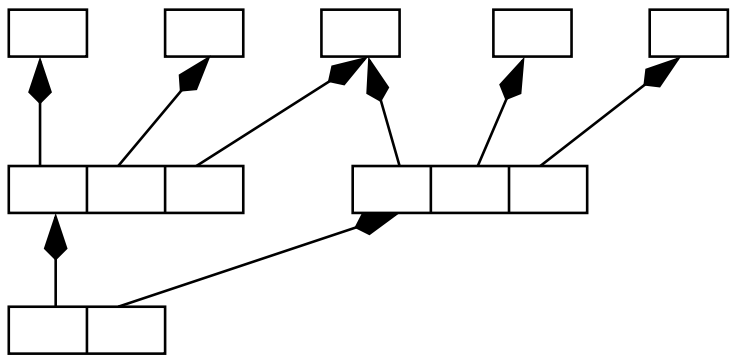
quasi-concurrency $\langle b, H\rangle qco \langle b', H'\rangle \iff$ H and H' are consistent and $\langle b, H\rangle$ is *read* in the history H' but not consumed.

Strutture Dati e Considerazioni sulla Complessità

- Definizione di regole per costruire incrementalmente le relazioni di concorrenza e quasi-concorrenza

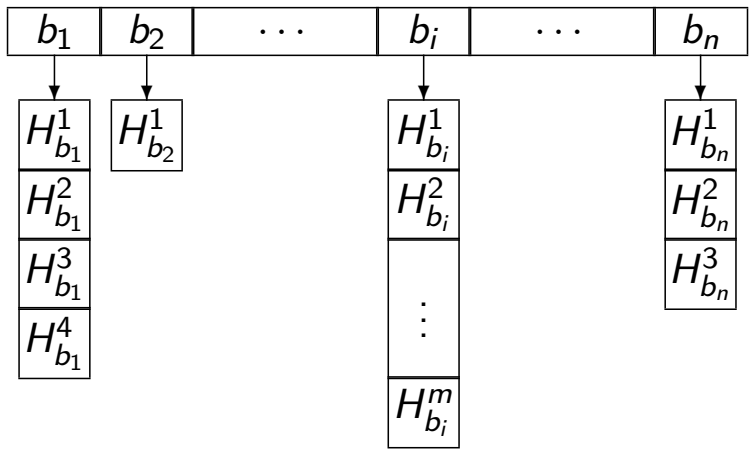
Strutture Dati e Considerazioni sulla Complessità

- Definizione di regole per costruire incrementalmente le relazioni di concorrenza e quasi-concorrenza
- Storie come liste di predecessori

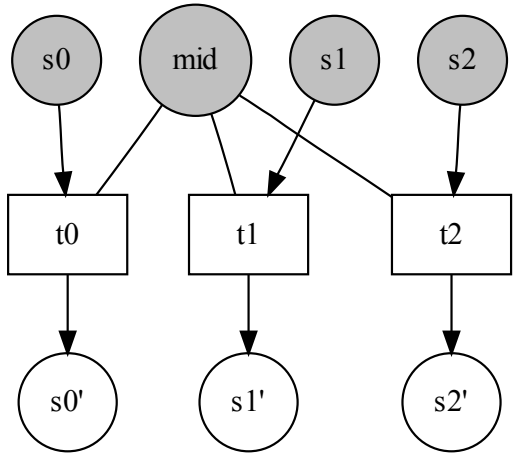


Strutture Dati e Considerazioni sulla Complessità

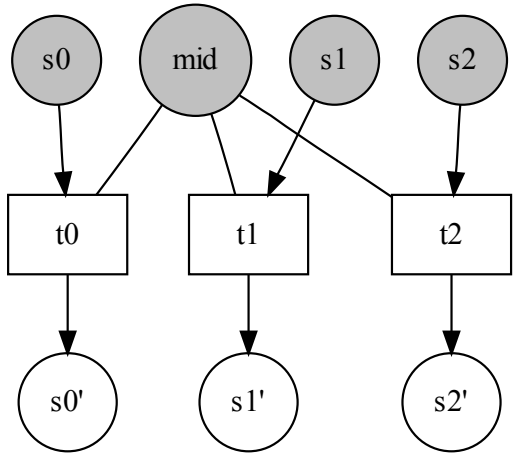
- Definizione di regole per costruire incrementalmente le relazioni di concorrenza e quasi-concorrenza
- Storie come liste di predecessori
- Gestione degli insiemi di coppie $\langle piazza, storia \rangle$ ed ordinamento

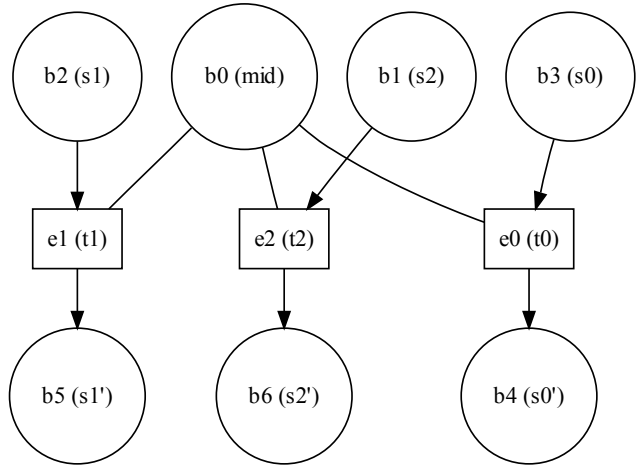


Esempio “Condizione nel Mezzo”

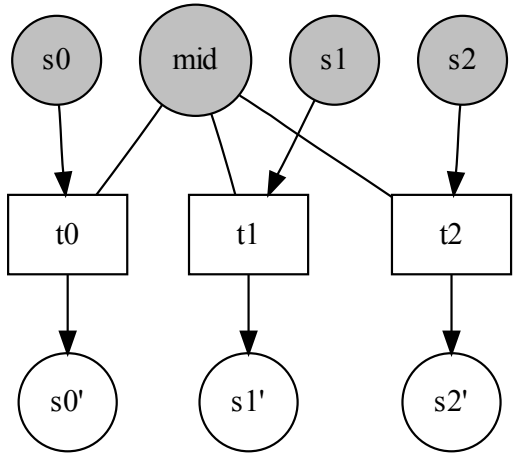


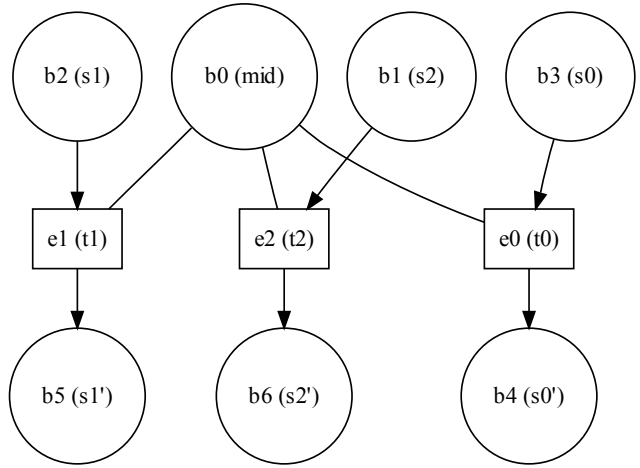
Esempio “Condizione nel Mezzo”





Esempio “Condizione nel Mezzo”

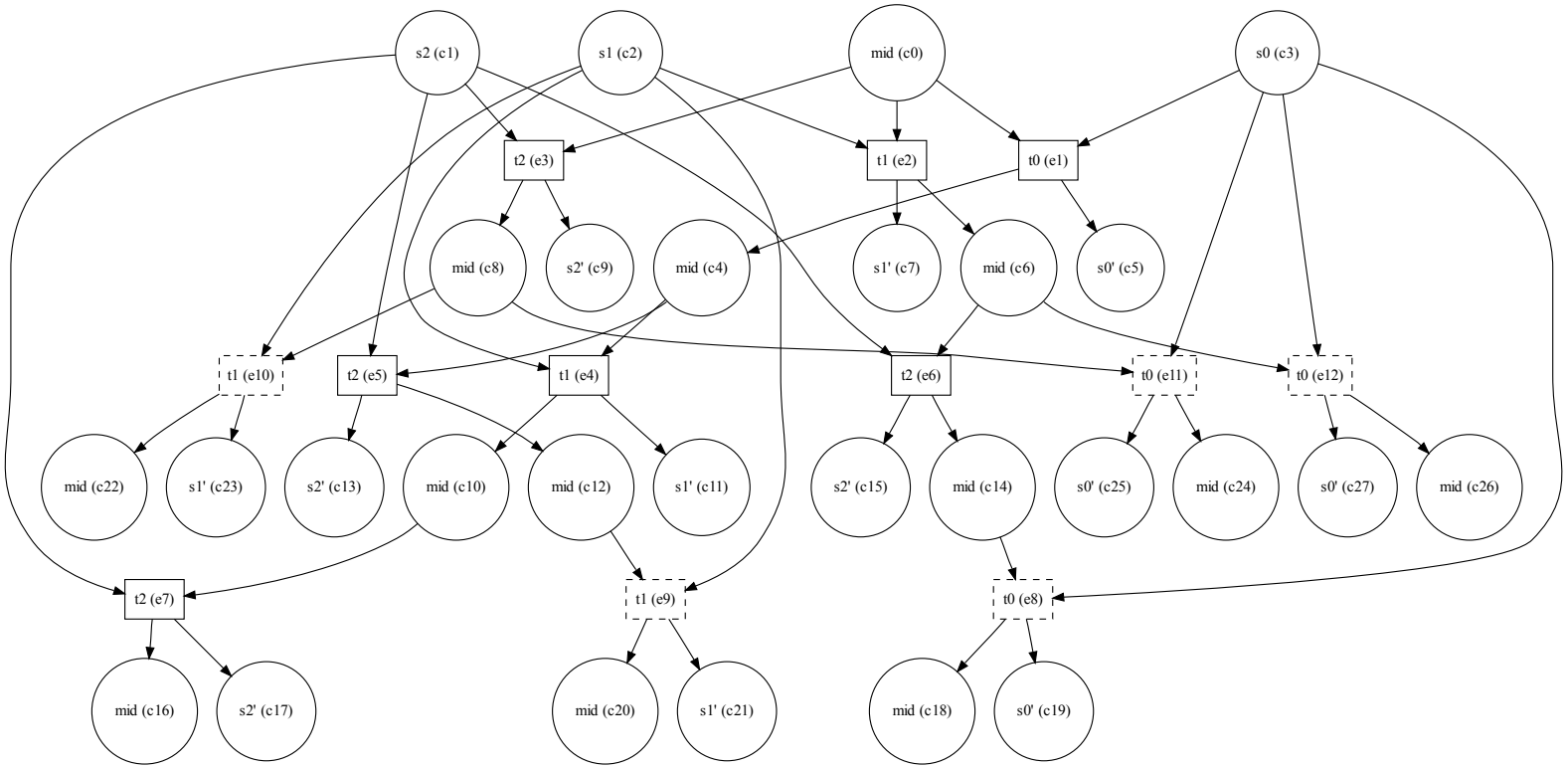




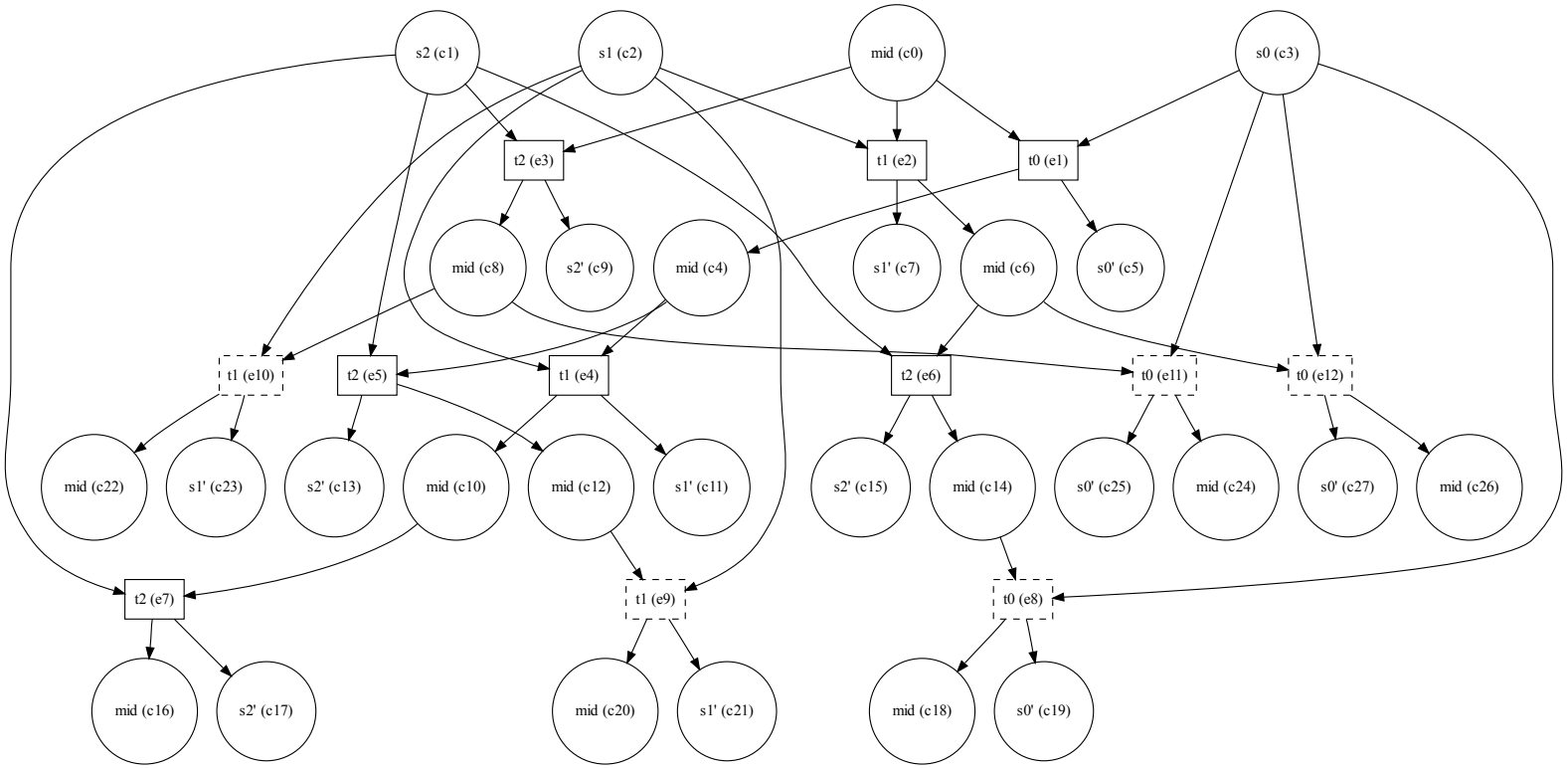
Storie rilevate:

- $H[e_0] = \{e_0\}$
- $H[e_1] = \{e_1\}$
- $H[e_2] = \{e_2\}$

Unfolding Prodotto da Mora

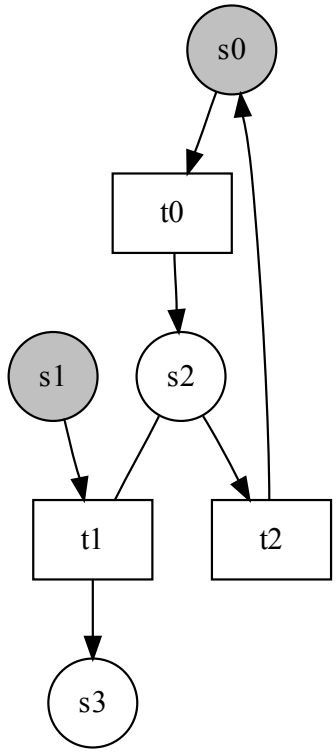


Unfolding Prodotto da Mora

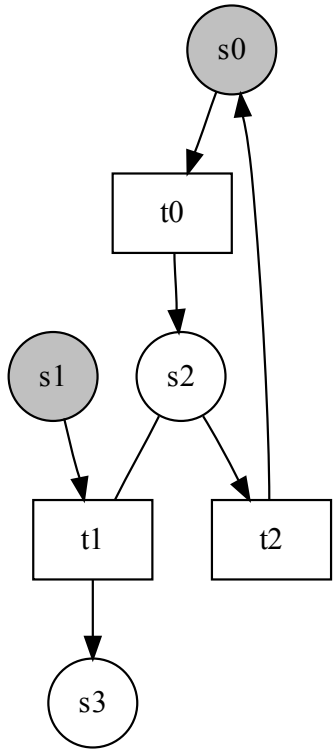


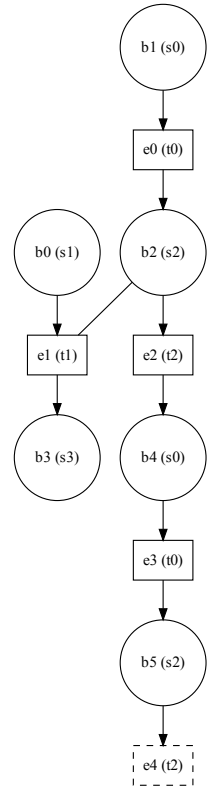
La struttura prodotta dal nuovo unfolder è esponenzialmente più piccola rispetto a quella prodotta da Mora.

Esempio con Ciclo

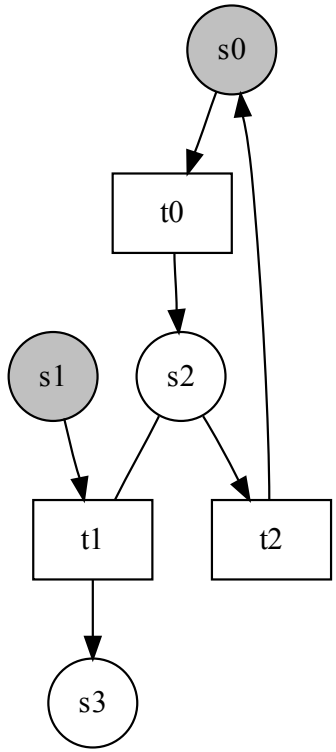


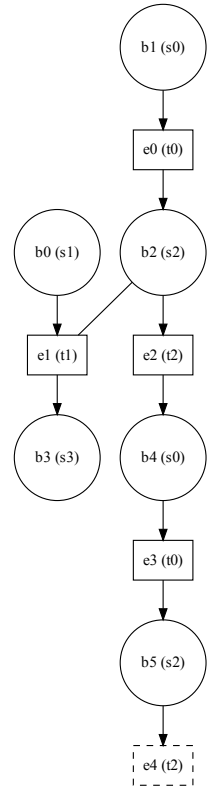
Esempio con Ciclo





Esempio con Ciclo





Le storie: $H[e_0] = \{e_0\}$, $H[e_1] = \{e_1, e_0\}$, $H[e_2] = \{e_2, e_1, e_0\}$, $H[e_3] = \{e_3, e_2, e_1, e_0\}$.

Strumenti e Sviluppo

- Tool esistente, Mora, sviluppato in C da Stefan Schwoon

Strumenti e Sviluppo

- Tool esistente, Mora, sviluppato in C da Stefan Schwoon
- Linguaggio e librerie: C e GLib2

Strumenti e Sviluppo

- Tool esistente, Mora, sviluppato in C da Stefan Schwoon
- Linguaggio e librerie: C e GLib2
  1. scelta di compatibilità col tool esistente

Strumenti e Sviluppo

- Tool esistente, Mora, sviluppato in C da Stefan Schwoon
- Linguaggio e librerie: C e GLib2
  1. scelta di compatibilità col tool esistente
  2. mancanza di template ed object orientation

Strumenti e Sviluppo

- Tool esistente, Mora, sviluppato in C da Stefan Schwoon
- Linguaggio e librerie: C e GLib2
  1. scelta di compatibilità col tool esistente
  2. mancanza di template ed object orientation
- Versionamento, build e documentazione
  1. CMake: build tool multipiattaforma e multiambiente
  2. Doxygen: strumento per la documentazione del codice
  3. Bazaar: versionamento (con Launchpad)

Strumenti e Sviluppo

- Tool esistente, Mora, sviluppato in C da Stefan Schwoon
- Linguaggio e librerie: C e GLib2
  1. scelta di compatibilità col tool esistente
  2. mancanza di template ed object orientation
- Versionamento, build e documentazione
  1. CMake: build tool multipiattaforma e multiambiente
  2. Doxygen: strumento per la documentazione del codice
  3. Bazaar: versionamento (con Launchpad)
- Debugging: GDB, Valgrind ed Electric Fence

Strumenti e Sviluppo

- Tool esistente, Mora, sviluppato in C da Stefan Schwoon
- Linguaggio e librerie: C e GLib2
  1. scelta di compatibilità col tool esistente
  2. mancanza di template ed object orientation
- Versionamento, build e documentazione
  1. CMake: build tool multipiattaforma e multiambiente
  2. Doxygen: strumento per la documentazione del codice
  3. Bazaar: versionamento (con Launchpad)
- Debugging: GDB, Valgrind ed Electric Fence
- Testing: utilizzo di asserzioni sul codice e test con esempi concreti

Tabella dei Requisiti

	Functional	
rf1	Compute the unfolding of a contextual Petri Net	Must
rf1.1	Using BCKS	Must
rf1.2	Using the already existing PR-encoding	May
rf2	Read a c-net in the form of a PEP LL-NET file	Must
rf3	Produce usable output	Must
rf3.1	Produce a dot-compatible output	Must
rf3.2	Save the output in PEP LL-NET format	Should
rf3.3	Convert an LL-NET to a dot file	May
	Quality	
rq1	Optimization of the algorithm	Must
rq1.1	Adequate orders	May
rq2	Parametrization of the algorithm	Must
	Constraint	
rc1	Integration with graphviz suite	Must
rc2	Reuse of the original tool	Must
rc2.1	Use of C programming language	Should
rc2.2	Be comparable with the original tool	Must
rc3	Cross-platform, at least POSIX compliant	Must
rc4	Documentation	Must

Tabella: Table of requirements.

Consumo Orario

Week	Task	Hours
1	Study of theoretical background	40
2	Study of theoretical background	40
3	Study of the existing tool	40
4	Design	30
5	Design	30
6	Design and implementation	40
7	Implementation	30
8	Implementation	30
9	Tests and refinement	40
	Total hours:	320

Tabella: Consumo orario e pianificazione settimanales

Conclusioni

- È stata realizzata un'implementazione — con particolare attenzione all'efficienza in termini di spazio e tempo — dell'algoritmo iniziale, rispettando tutti i requisiti obbligatori e buona parte degli opzionali

Conclusioni

- È stata realizzata un'implementazione — con particolare attenzione all'efficienza in termini di spazio e tempo — dell'algoritmo iniziale, rispettando tutti i requisiti obbligatori e buona parte degli opzionali
- Rimangono comunque alcune questioni aperte:

Conclusioni

- È stata realizzata un'implementazione — con particolare attenzione all'efficienza in termini di spazio e tempo — dell'algoritmo iniziale, rispettando tutti i requisiti obbligatori e buona parte degli opzionali
- Rimangono comunque alcune questioni aperte:
  1. Applicazione e studio di ordini adeguati

Conclusioni

- È stata realizzata un'implementazione — con particolare attenzione all'efficienza in termini di spazio e tempo — dell'algoritmo iniziale, rispettando tutti i requisiti obbligatori e buona parte degli opzionali
- Rimangono comunque alcune questioni aperte:
  1. Applicazione e studio di ordini adeguati
  2. Confronto e integrazione di altri algoritmi (es. Mora)

Conclusioni

- È stata realizzata un'implementazione — con particolare attenzione all'efficienza in termini di spazio e tempo — dell'algoritmo iniziale, rispettando tutti i requisiti obbligatori e buona parte degli opzionali
- Rimangono comunque alcune questioni aperte:
  1. Applicazione e studio di ordini adeguati
  2. Confronto e integrazione di altri algoritmi (es. Mora)
  3. Studio teorico e dimostrazione formale degli algoritmi per il calcolo del prefisso finito e completo

Conclusioni

- È stata realizzata un'implementazione — con particolare attenzione all'efficienza in termini di spazio e tempo — dell'algoritmo iniziale, rispettando tutti i requisiti obbligatori e buona parte degli opzionali
- Rimangono comunque alcune questioni aperte:
  1. Applicazione e studio di ordini adeguati
  2. Confronto e integrazione di altri algoritmi (es. Mora)
  3. Studio teorico e dimostrazione formale degli algoritmi per il calcolo del prefisso finito e completo
  4. Applicazione di algoritmi di verifica sul prefisso generato

Conclusioni

- È stata realizzata un'implementazione — con particolare attenzione all'efficienza in termini di spazio e tempo — dell'algoritmo iniziale, rispettando tutti i requisiti obbligatori e buona parte degli opzionali
- Rimangono comunque alcune questioni aperte:
  1. Applicazione e studio di ordini adeguati
  2. Confronto e integrazione di altri algoritmi (es. Mora)
  3. Studio teorico e dimostrazione formale degli algoritmi per il calcolo del prefisso finito e completo
  4. Applicazione di algoritmi di verifica sul prefisso generato
  5. Applicazione dell'approccio ad altri formalismi più espressivi, come i sistemi di riscrittura su grafi