Ingegneria dei requisiti

Requisiti

Proprietà (funzionali e non) che l'applicazione dovrà avere, descrivono

CHE COSA il sistema dovrà fare piuttosto che COME lo dovrà fare,

sono focalizzati sul

PROBLEMA non sulla SOLUZIONE

(Vedi capitolo 5 Sommerville)

Elicitazione e analisi dei requisiti

- L'elicitazione dei requisiti è una attività critica per la riuscita del progetto: errori introdotti in questa fase hanno un costo enorme se scoperti troppo tardi
- I requisiti sono uno strumento molto importante di comunicazione con il committente
- L'analisi ha luogo quando l'esperto del dominio è presente, altrimenti è pseudoanalisi
- Uno degli elementi più importanti quando si gestiscono i rischi legati ai requisiti è avere accesso alla conoscenza degli esperti del dominio
- La mancanza di contatto con gli esperti è una delle cause più comuni di fallimento di un progetto

(Vedi capitolo 6 Sommerville)

Caratteristiche dei requisiti

1. Validità – ogni requisito esprime qualcosa di cui l'utente ha realmente bisogno
2. Correttezza – la descrizione di ciascun requisito non contiene errori
3. Consistenza – non ci sono conflitti tra i requisiti
4. Completezza (esterna e interna) – tutti gli stati, le funzionalità, gli input, gli output e i vincoli sono contemplati da qualche requisito
5. Realismo – effettiva realizzabilità
6. Comprensibilità
7. Tracciabilità – riconducibilità alle ragioni / origini
8. Modificabilità
9. Verificabilità – quando si formula un requisito, si deve stabilire come decidere a posteriori se esso è stato realizzato dal sistema oppure no

Specifica dei requisiti

Tipologie di applicazioni (o di parti di esse)	Requisiti da specificare
Sequenziali (unico flusso di controllo)	Funzionalità offerte
Concorrenti (più flusso paralleli di controllo + meccanismi di sincronizzazione per l'accesso a risorse condivise)	Attività parallele, risorse condivise, meccanismi di sincronizzazione
Dipendenti dal tempo (dette "in tempo reale": la correttezza dei risulti dipende dal tempo di esecuzione delle attività)	Come per le attività concorrenti + tempi di esecuzione

Linguaggi di specifica

NON esiste un linguaggio di specifica adatto per qualunque problema o classe di applicazioni

Ulteriore categorizzazione delle applicazioni (o di parti di esse)	Requisiti da specificare	Linguaggi di specifica
Orientate ai dati (es. sistemi informativi)	Struttura concettuale dei dati	Modello ER
Orientate alle funzioni (es. traduttori)	Funzioni e flusso dei dati	Diagrammi di flusso (DFD)
Orientate al controllo (es. applicazioni dipendenti dal tempo che interagiscono con l'ambiente esterno)	Attività parallele, risorse condivise, tempi di esecuzione	Automi a stati finiti, reti di Petri, diagrammi di stato UML (statechart)

Linguaggi di specifica (cont.)

Specifiche	Linguaggi di specifica	Note
Informali	linguaggio naturale (che è impreciso, ambiguo e ridondante)	
Formali: in formalismo matematico (che obbliga alla precisione, è passibile di manipolazione automatica, talvolta consente di provare automaticamente consistenza e completezza, può essere eseguito)	Z (sequenziale), Reti di Petri (concorrente)	Una specifica eseguibile costituisce un prototipo → può essere convalidata da committenti/utenti
Semiformali: notazioni (più o meno) rigorose, spesso grafiche + annotazioni in linguaggio naturale	ER, DFD, casi d'uso + diagramma delle classi (+ diagramma delle attività) UML	

Linguaggi di specifica (cont.)

Notazione =
sintassi del linguaggio di modellazione (aspetto grafico nel caso di linguaggi grafici; la sintassi UML è descritta da un meta-modello in UML)

Notazioni semiformali:
il loro significato non è univoco, bensì viene lasciato all'interpretazione dell'utente o dello strumento CASE generatore di codice (es. Together)

Documento dei requisiti in linguaggio naturale

I requisiti sono organizzati e numerati univocamente (dot notation) per categorie generali, suddivise ricorsivamente in sottocategorie, fino ad arrivare ai singoli requisiti atomici. La decomposizione riflette la prospettiva del cliente circa il sistema commissionato

Esempio:

1 Categoria
1.1 Sottocategoria
1.1.1 Requisito 1 della sottocategoria 1 della categoria 1
1.1.2 Requisito 2 della sottocategoria 1 della categoria 1
2 Categoria
……

Documento dei requisiti in linguaggio naturale (cont.)

Ogni azienda si dota del proprio formato per il documento dei requisiti, che tipicamente, per ogni requisito, include le seguenti informazioni:

1. Numero identificatore unico (es. 1. 2. 5)
2. Descrizione sintetica (una riga, es. stampa fattura)
3. Descrizione dettagliata
4. Ingressi: tipo e provenienza
5. Uscite: tipo e destinazione
6. Interazioni con l'utente
7. Altre entità coinvolte
8. Pre-condizioni
9. Post-condizioni
10. Effetti collaterali
11. Requisiti attinenti (see also…)

Documento dei requisiti in linguaggio naturale (cont.)

Esempio (1).

ID 3.5.1	
Function	Add node
Description	Adds a node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.
Inputs	Node type, Node position, Design identifier.
Source	Node type and Node position are input by the user, Design identifier from the database.
Outputs	Design identifier.
Destination	The design database. The design is committed to the database on completion of the operation.

Documento dei requisiti in linguaggio naturale (cont.)

Esempio (2).

Requires	Design graph rooted at input design identifier.
Pre-condition	The design is open and displayed on the user's screen.
Post-condition	The design is unchanged apart from the addition of a node of the specified type at the given position.
Side-effects	None

Gestione dei requisiti nel RUP

Fase di elaborazione: scoperta dei casi d'uso

Scoperta, mediante interviste con gli utenti, del maggior numero possibile di casi d'uso, in particolare quelli più importanti e quelli connessi a rischi maggiori e descrizione degli stessi mediante brevi testi scritti in linguaggio naturale

Fase di elaborazione: analisi del dominio

Costruzione, mediante l'interazione con gli esperti e guidata anche dai casi d'uso via via che emergono, di un **modello** (concettuale) **del dominio** che:

Gestione dei requisiti nel RUP (cont.)

(a) risponde a queste domande:
- Quali sono i termini specifici del dominio?
- Qual è il loro significato?
- Quali sono i concetti chiave?
- Come interagiscono le varie entità fra di loro?

(b) descrive il MONDO che circonda l'applicazione sw che si svilupperà (e che sarà supportato da essa)

(c) è comprensibile agli esperti del dominio, che devono poterlo discutere con gli analisti

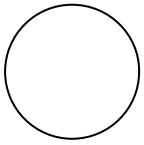
(d) NON è un modello di alto livello di astrazione (ovvero un modello in cui sono omessi molti dettagli) ma piuttosto lo scheletro (ovvero è un modello dettagliato ma solo di una piccola parte del tutto, quella che è fondamento del resto del modello)

Diagramma di flusso dati (DFD)

- Sono focalizzati su operazioni e flussi di informazione
- Incarnano il concetto di raffinamento della specifica (o astrazione mediante incapsulamento) effettuando la scomposizione di un'operazione in operazioni più semplici → incrementalità nella produzione della specifica

Elementi grafici

Agente esterno,
Interazione di I/O



Funzione o Processo

Flusso di Dati

Deposito (permanente) di Dati

Diagramma di flusso dati (cont.)

L’esempio di una biblioteca

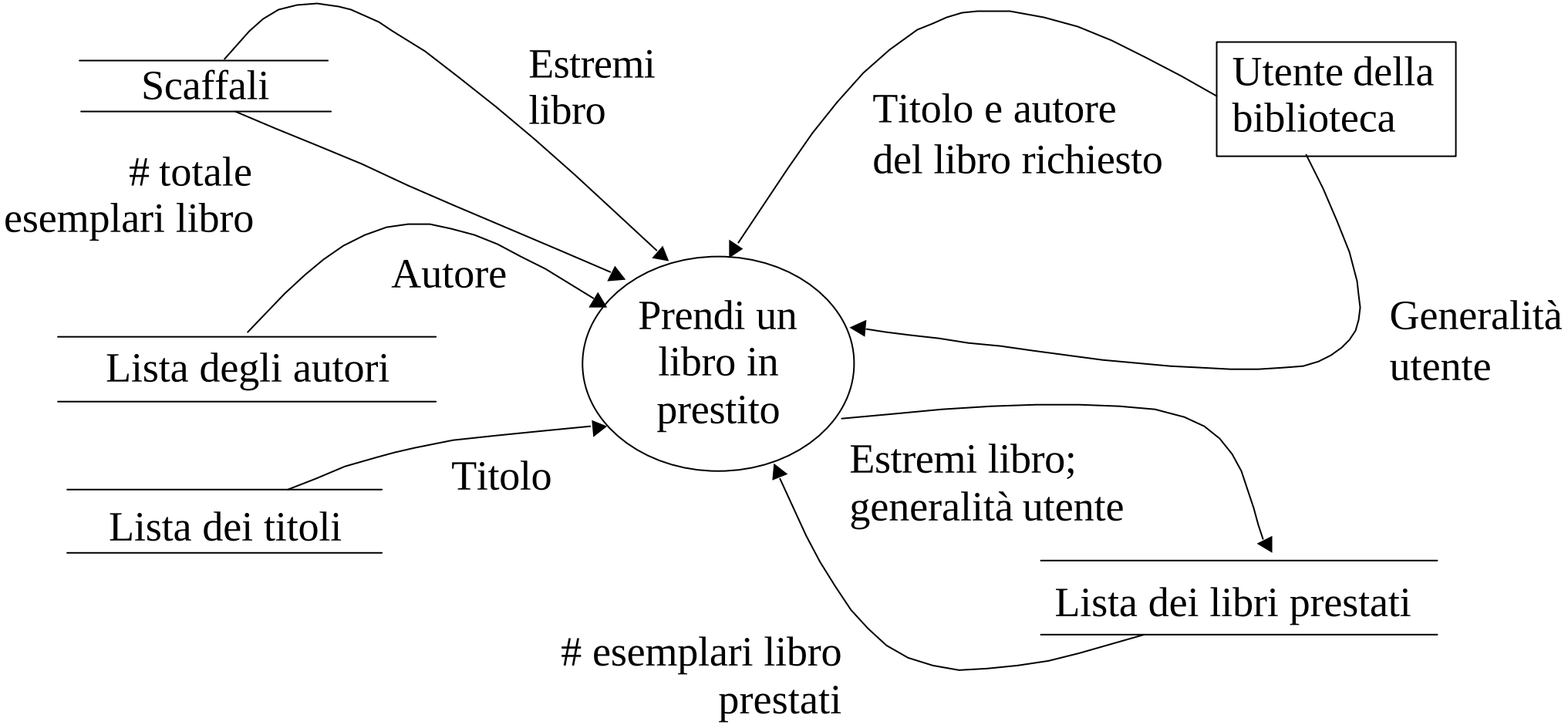


Diagramma di flusso dati (cont.)

Vincolo della continuità del flusso informativo:
nel diagramma corrispondente a una funzione che è stata esplosa, i flussi in entrata e uscita devono essere gli stessi flussi della funzione originale (mentre si possono introdurre nuovi agenti e depositi “locali”)

Diagramma di flusso dati (cont.)

Raffinamento della funzione “Prendi un libro in prestito”

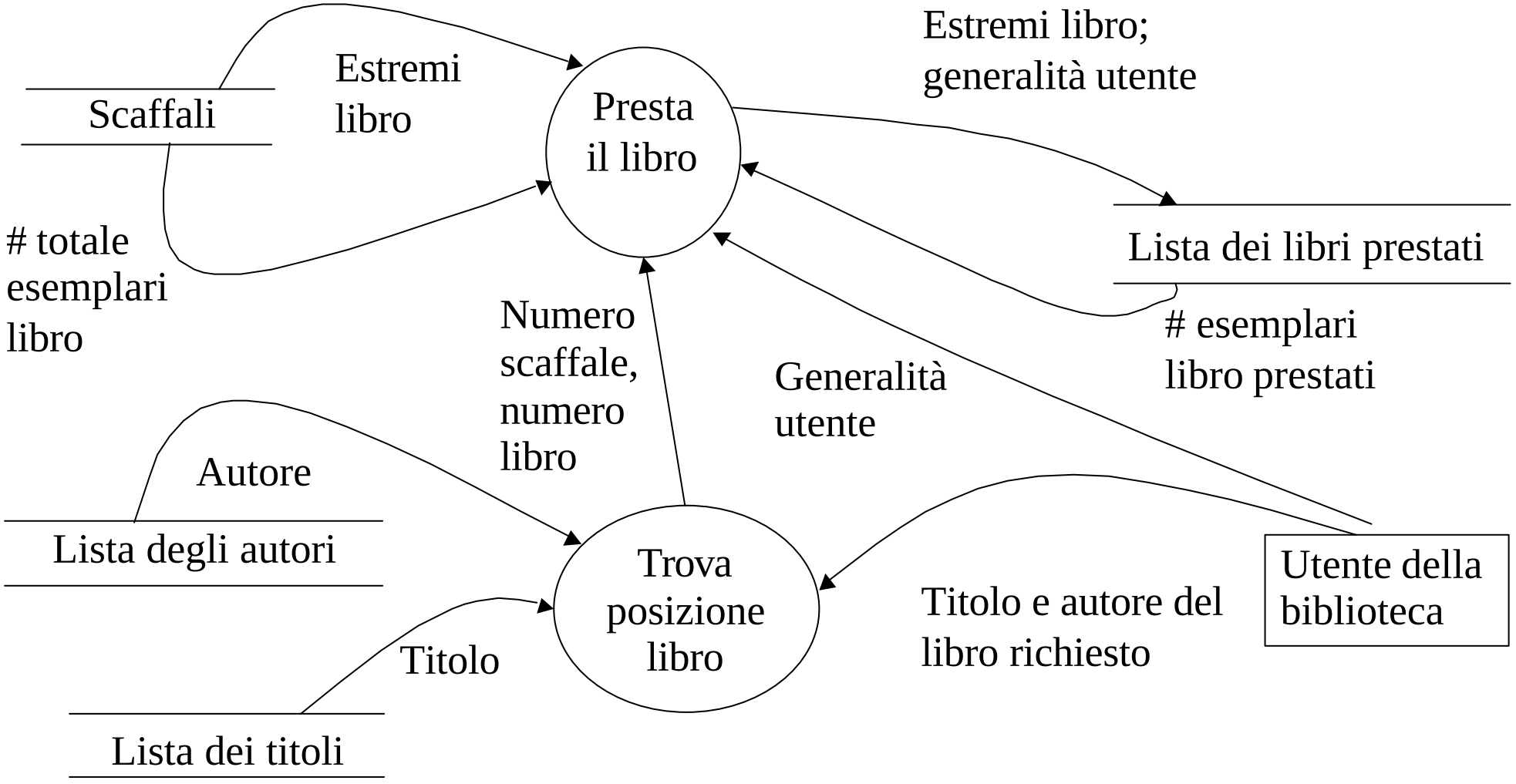


Diagramma di flusso dati (cont.)

Linee guida per la costruzione di un DFD:

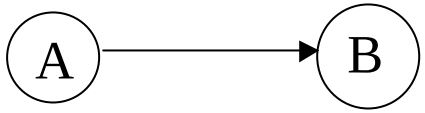
- ignorare inizializzazione, terminazione e casi di errore; concentrarsi sulle condizioni stabili
- ignorare flusso di controllo e sincronizzazione
- individuare innanzi tutto ingressi e uscite esterni al (sotto) sistema che si sta descrivendo
- usare nomi significativi
- percorrere i flussi informativi per valutare la correttezza e la consistenza del diagramma

Pregio dei DFD: sono facili da leggere

Limiti dei DFD:

- non consentono di esprimere il controllo di flusso né le sincronizzazioni
- sono ambigui

Diagramma di flusso dati: ambiguità

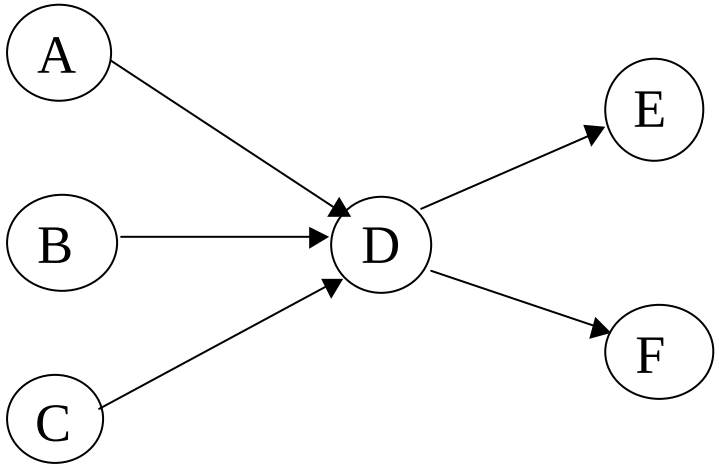
Es. 

Possibili interpretazioni:

(a) A produce un dato e aspetta finché B lo consuma

(b) B può leggere il dato più volte senza consumarlo

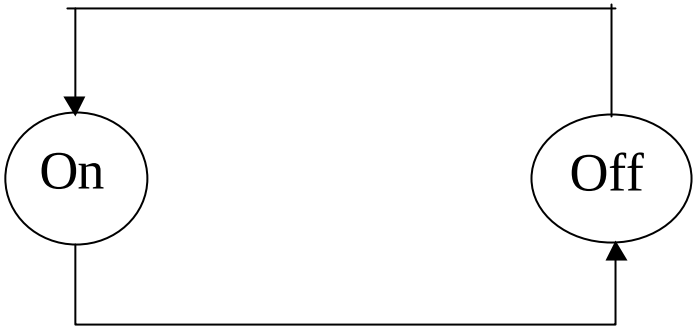
(c) A e B sono connessi mediante un meccanismo a “pipe”



- Le uscite di A, B, e C sono tutte sempre necessarie ?
- Le uscite inviate a E e F sono uguali, oppure sono prodotte contemporaneamente, oppure sono prodotte alternativamente ?

Macchine a stati finiti (FSM)

S: insieme finito di stati
I: insieme finito degli ingressi
δ: funzione di transizione

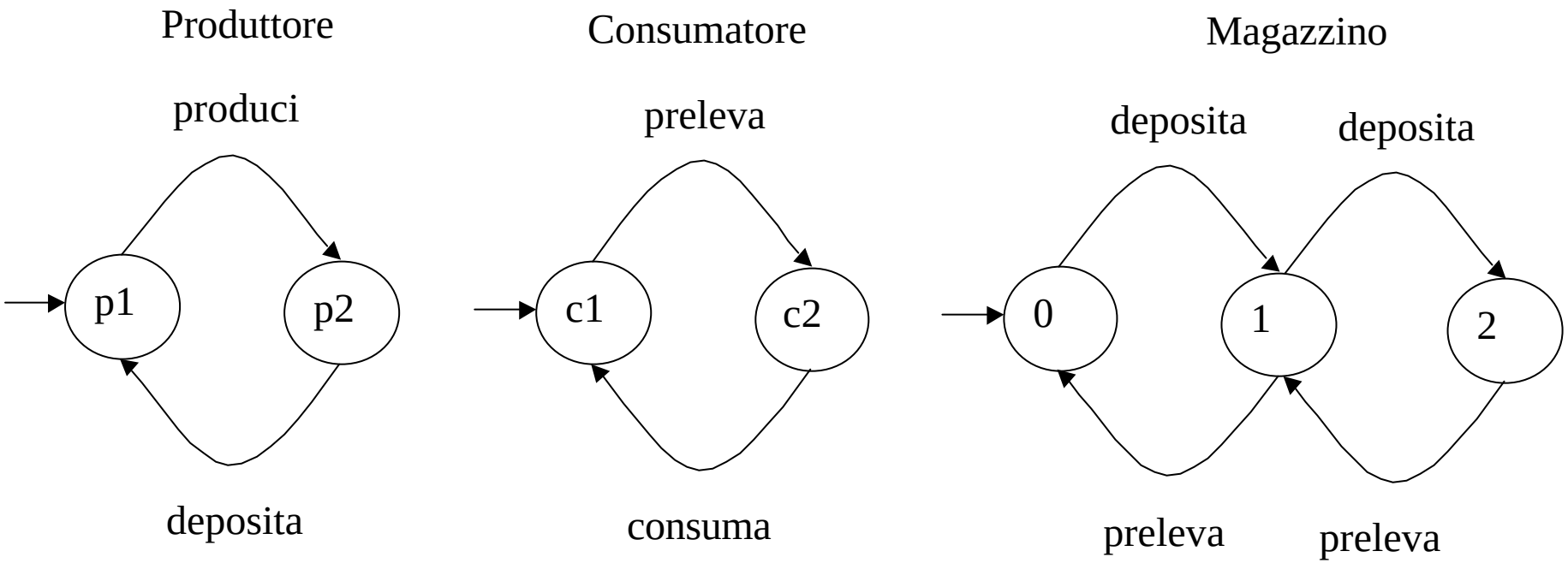


Classi di FSM

- Deterministiche/nondeterministiche
- Riconoscitori
- Traduttori

Macchine a stati finiti (cont.)

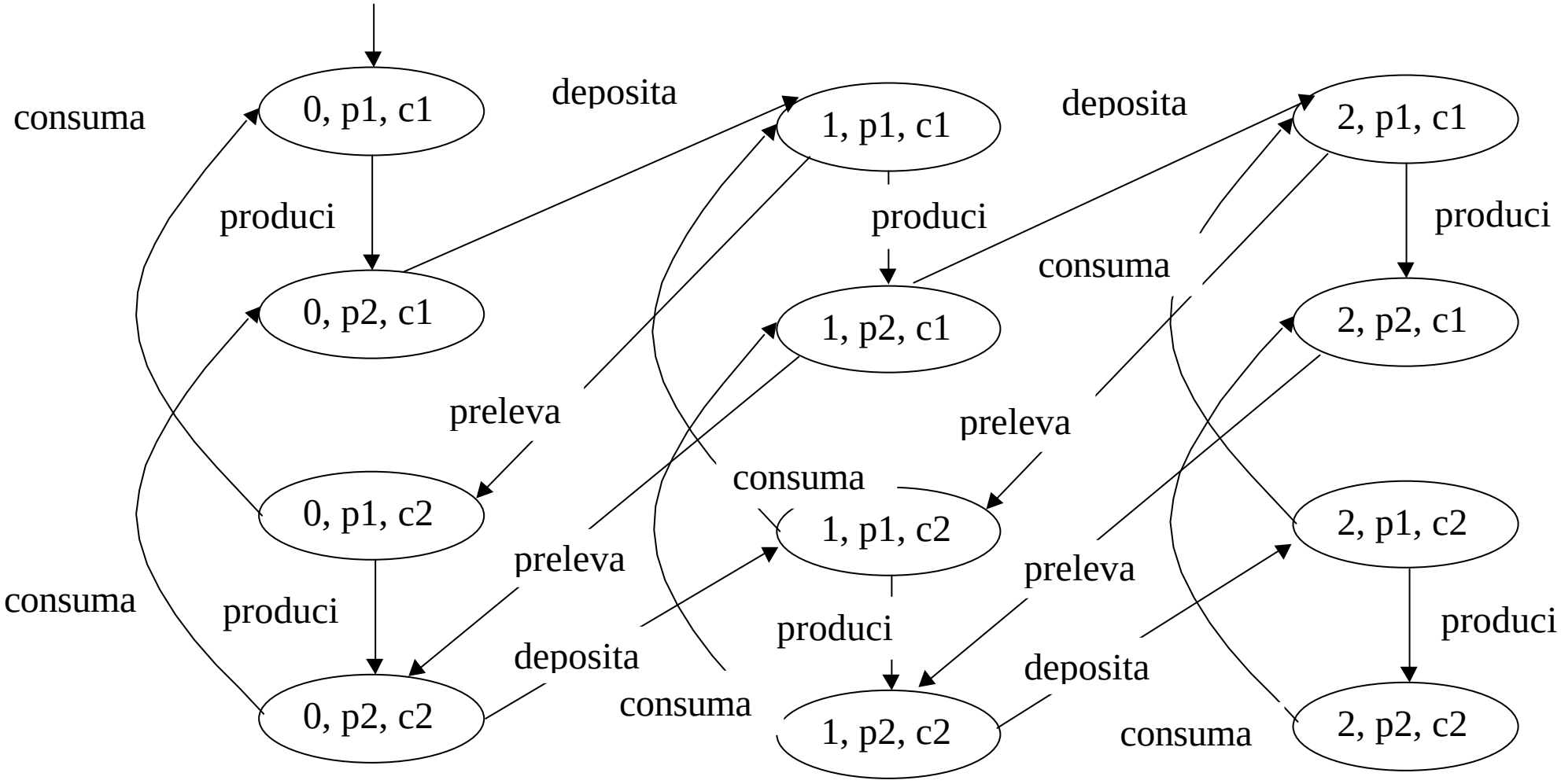
Esempio: produttore-consumatore



Limitazione

Esplosione degli stati quando si rappresentano sistemi concorrenti: date n FSM con $k1, k2, ..., k_n$ stati, la loro composizione è una FSM con $k1 * k2 * ... * k_n$ stati (mentre vorremmo che fossero $k1 + k2 + ... + k_n$)

Macchine a stati finiti (cont.)

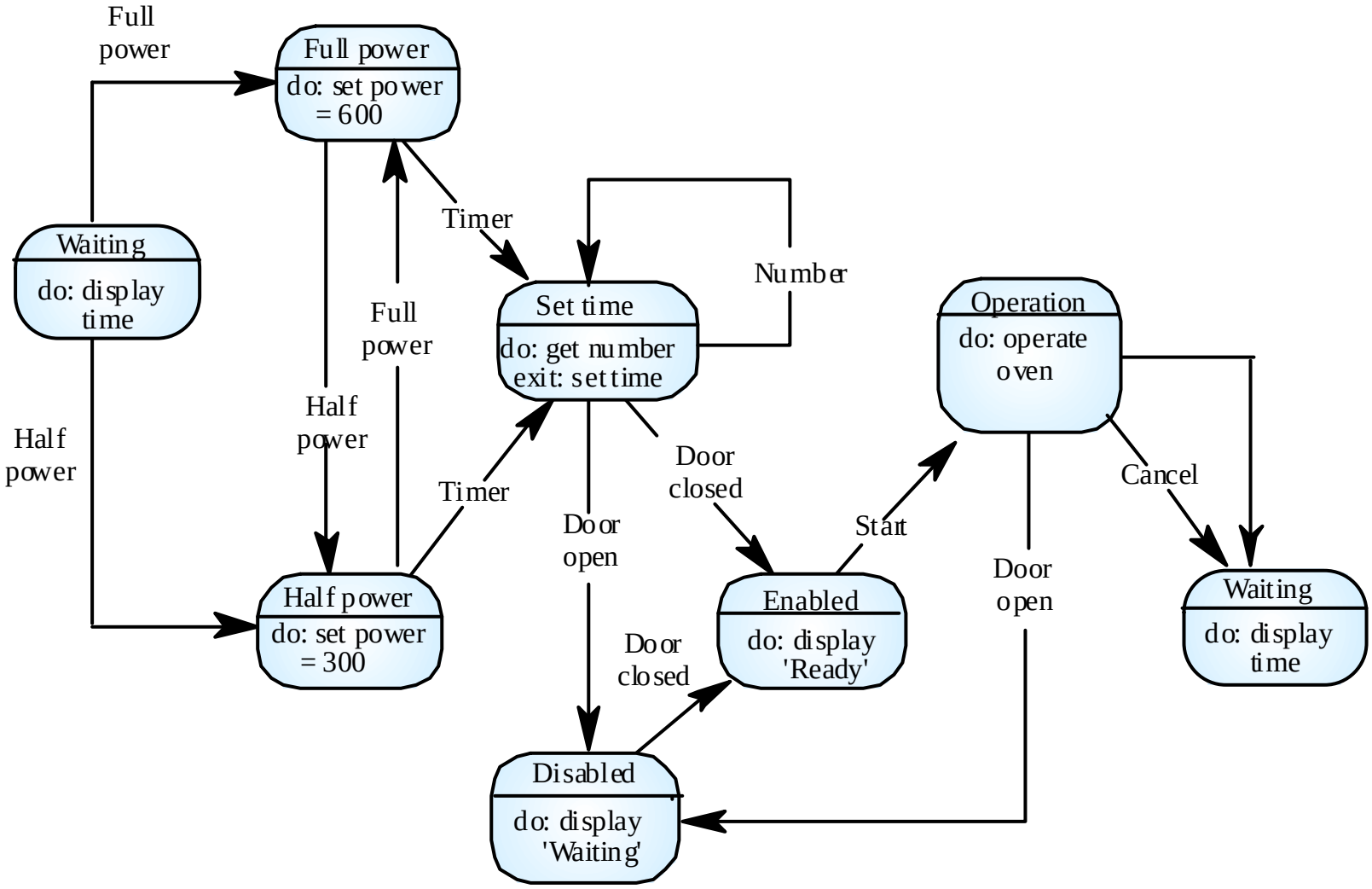


Macchine a stati finiti (cont.)

Soluzione della limitazione delle FSM:

- Statechart (FSM cooperanti, usate in UML)
- Reti di Petri

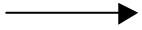
Statechart


Full power
Full power
do: set power = 600
Waiting
do: display time
Timer
Number
Set time
do: get number
exit: set time
Operation
do: operate oven
Full power
Half power
Half power
Timer
Door closed
Door open
Start
Cancel
Door open
Half power
do: set power = 300
Enabled
do: display 'Ready'
Waiting
do: display time
Door closed
Disabled
do: display 'Waiting'

Reti di Petri

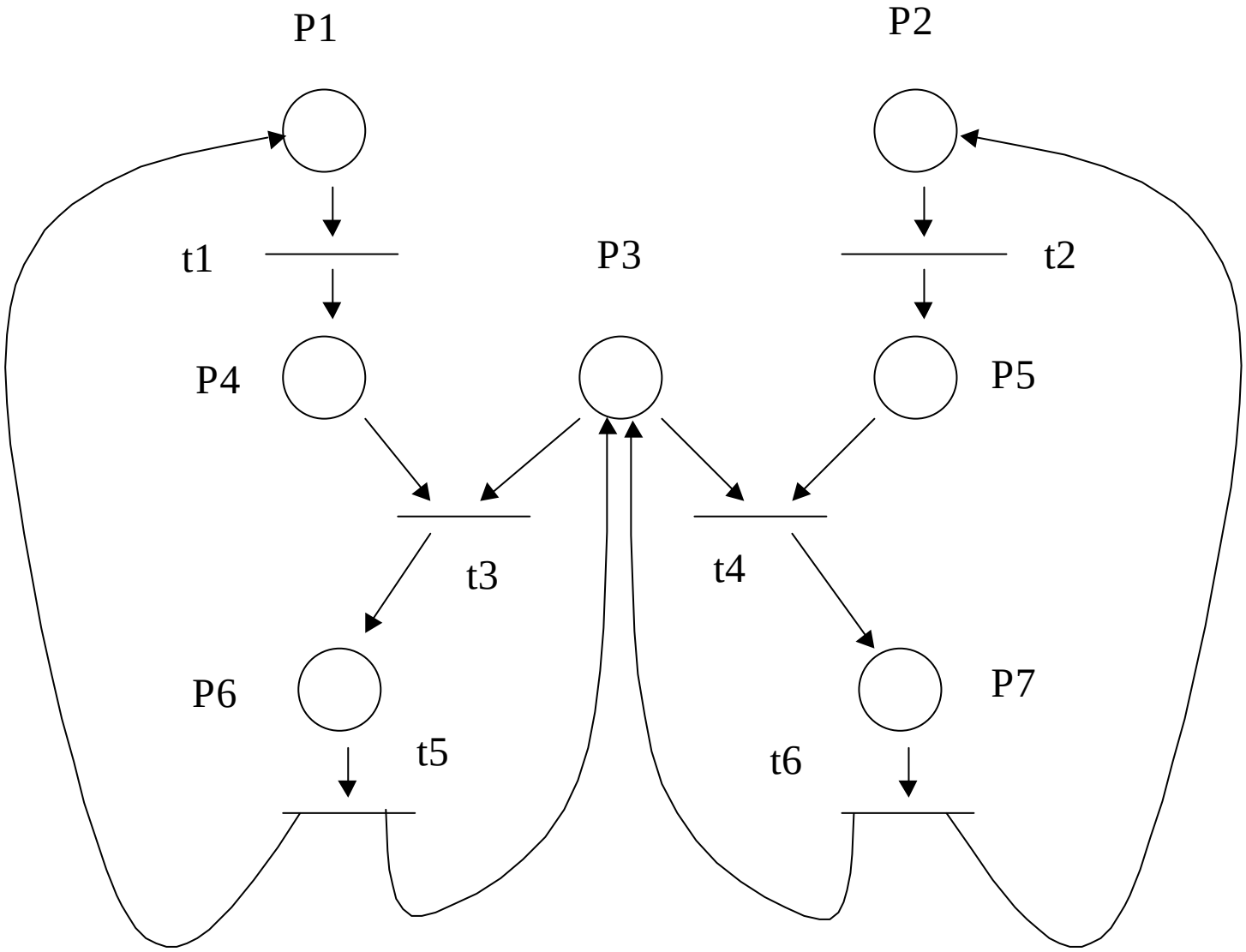
- I concetti di stato e transizione non sono più centralizzati ma distribuiti
- Non esiste una frequenza di "clock" → descrizione naturale di sistemi asincroni
- L'evoluzione è non deterministica

N = (P, T, F)

con
P: insieme finito di posti (places) 
T: insieme finito di transizioni 
F: relazione di flusso 

Vincoli:
(1) $P \cap T = \varnothing$
(2) $P \cup T \neq \varnothing$
(3) $F \subseteq (P \times T) \cup (T \times P)$

Reti di Petri (cont.)



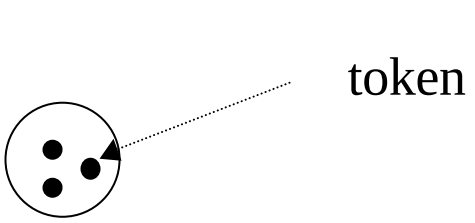
Reti di Petri (cont.)

Pre (y) = { x ∈ P ∪ T | ⟨ x, y ⟩ ∈ F }
Post (x) = { y ∈ P ∪ T | ⟨ x, y ⟩ ∈ F }

Modello di base: N = (P, T, F, W, M_0)

W: peso
M_0 : marcatura iniziale

Vincoli:
(4) W: F → *N* - {0}, valore di default di W è 1
(5) M_0: P → *N* (un funzione M: P → *N* è chiamata *marcatura*)

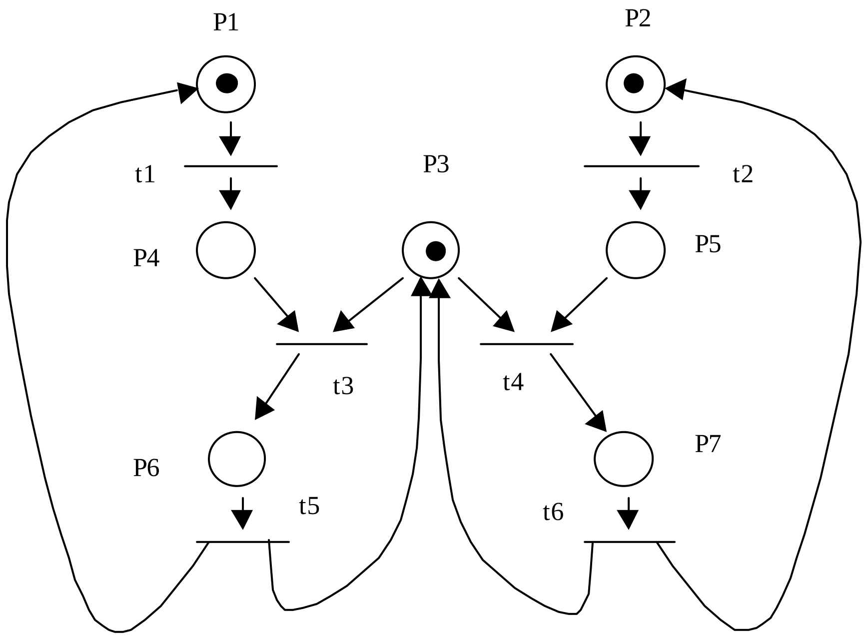


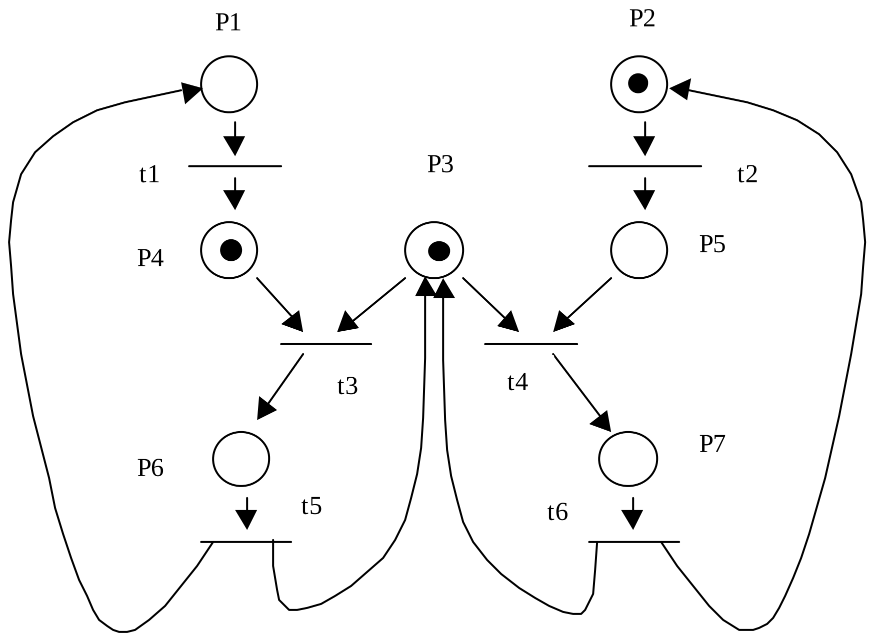
- Una marcatura è una rappresentazione di uno "stato" della rete
- La *semantica* definisce l'insieme delle possibili marcature prossime M', data la marcatura corrente M
- *Evoluzione dinamica*: $M_0 \Rightarrow M_1 \Rightarrow M_2 \Rightarrow ...$

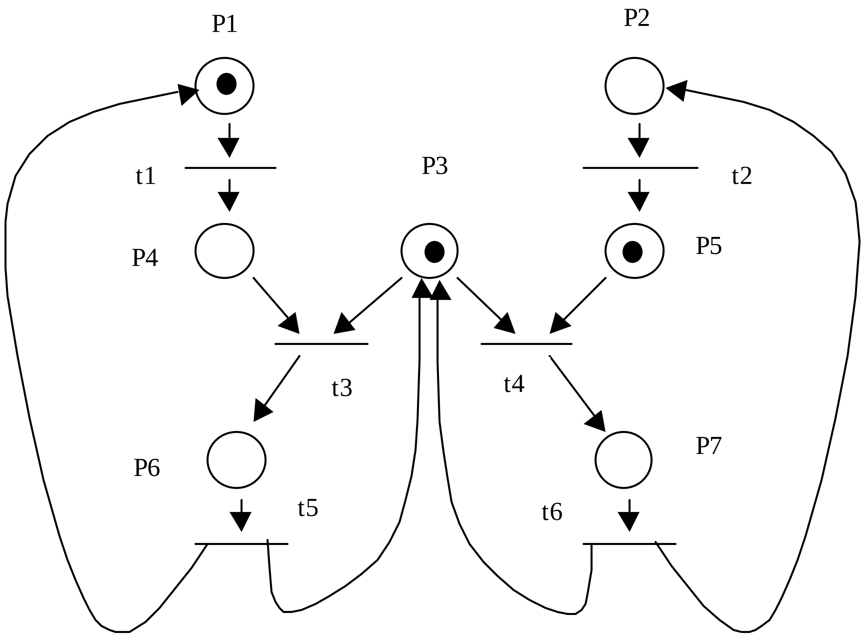
Reti di Petri (cont.)

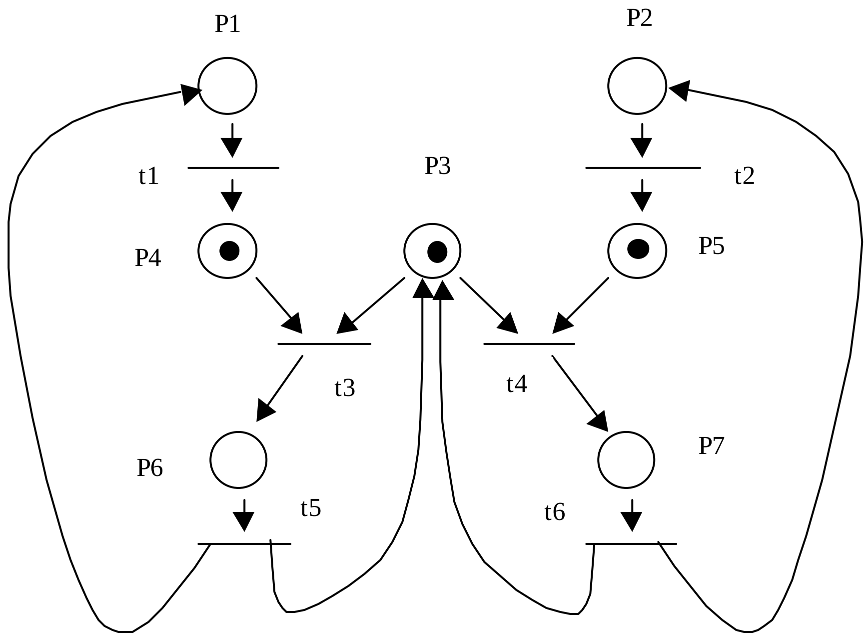
Semantica: descrizione informale (caso W=1)

- una transizione t è *abilitata* se c'è (almeno) un token in ogni p ∈ Pre(t)
- una transizione t abilitata può scattare, dando luogo a una nuova marcatura dove:
  - un token è cancellato da ogni p ∈ Pre(t)
  - un token è aggiunto a ogni p ∈ Post(t)
- se esistono più transizioni abilitate uscenti dallo stesso posto, la scelta della transizione da eseguire è nondeterministica


P1
P2
t1
P3
t2
P4
P5
t3
t4
P6
P7
t5
t6


P1
P2
t1
P3
t2
P4
P5
t3
t4
P6
P7
t5
t6


P1
P2
t1
P3
t2
P4
P5
t3
t4
P6
P7
t5
t6


P1
P2
t1
P3
t2
P4
P5
t3
t4
P6
P7
t5
t6

Reti di Petri (cont.)

Semantica: descrizione formale (caso generale)

- Abilitazione di una transizione t data in corrispondenza della marcatura M (si scrive M[t>):

$$\forall p \in \text{Pre}(t) \bullet M(p) \geq W(\langle p, t \rangle)$$

- Scatto di una transizione t (M[t > M'):

$$\forall p \in \text{Pre}(t) - \text{Post}(t) \bullet M'(p) = M(p) - W(\langle p, t \rangle)$$

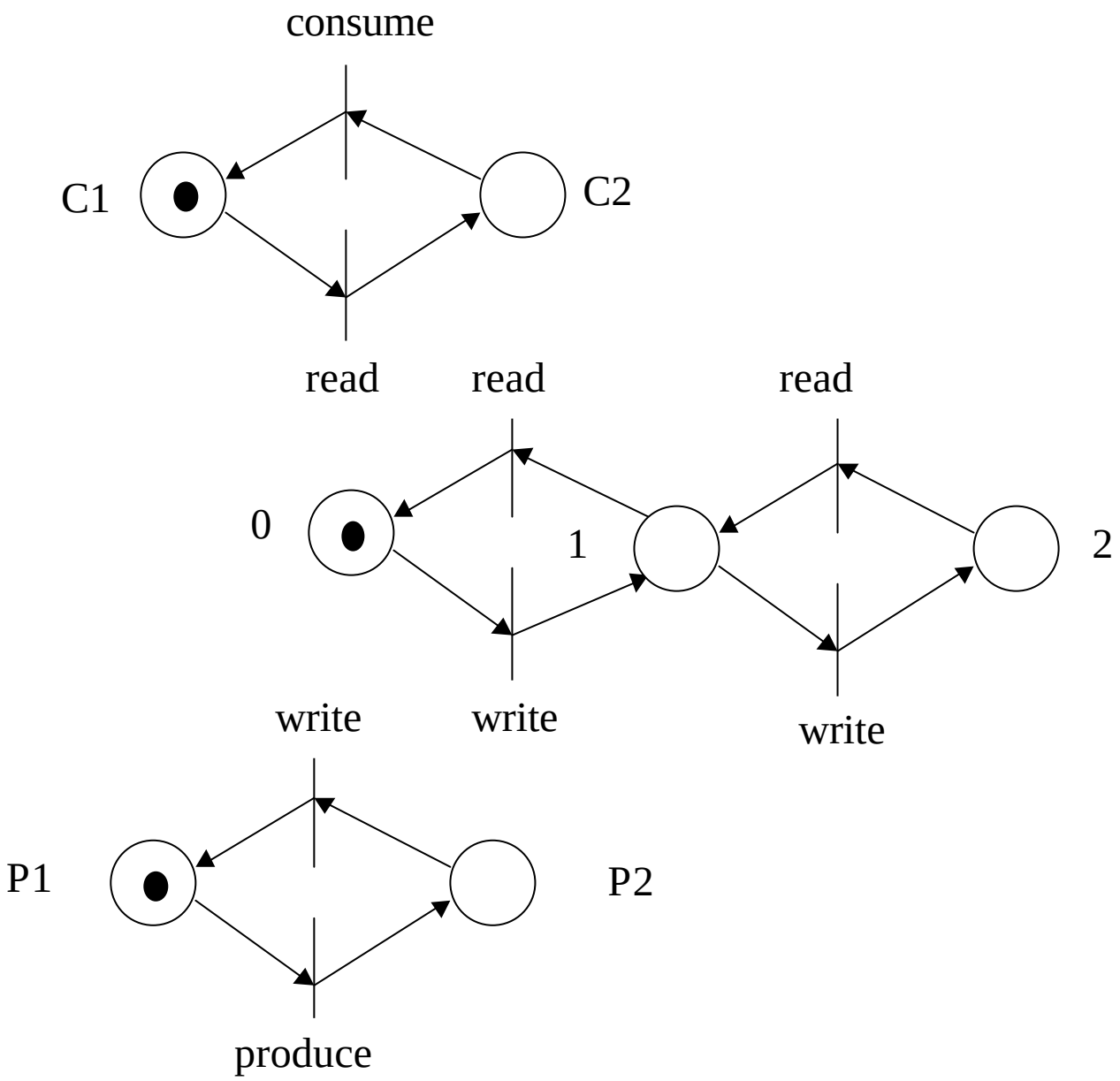
$$\forall p \in \text{Post}(t) - \text{Pre}(t) \bullet M'(p) = M(p) + W(\langle t, p \rangle)$$

$$\forall p \in \text{Pre}(t) \cap \text{Post}(t) \bullet M'(p) = M(p) - W(\langle p, t \rangle) + W(\langle t, p \rangle)$$

$$\forall p \in P - (\text{Pre}(t) \cup \text{Post}(t)) \bullet M'(p) = M(p)$$

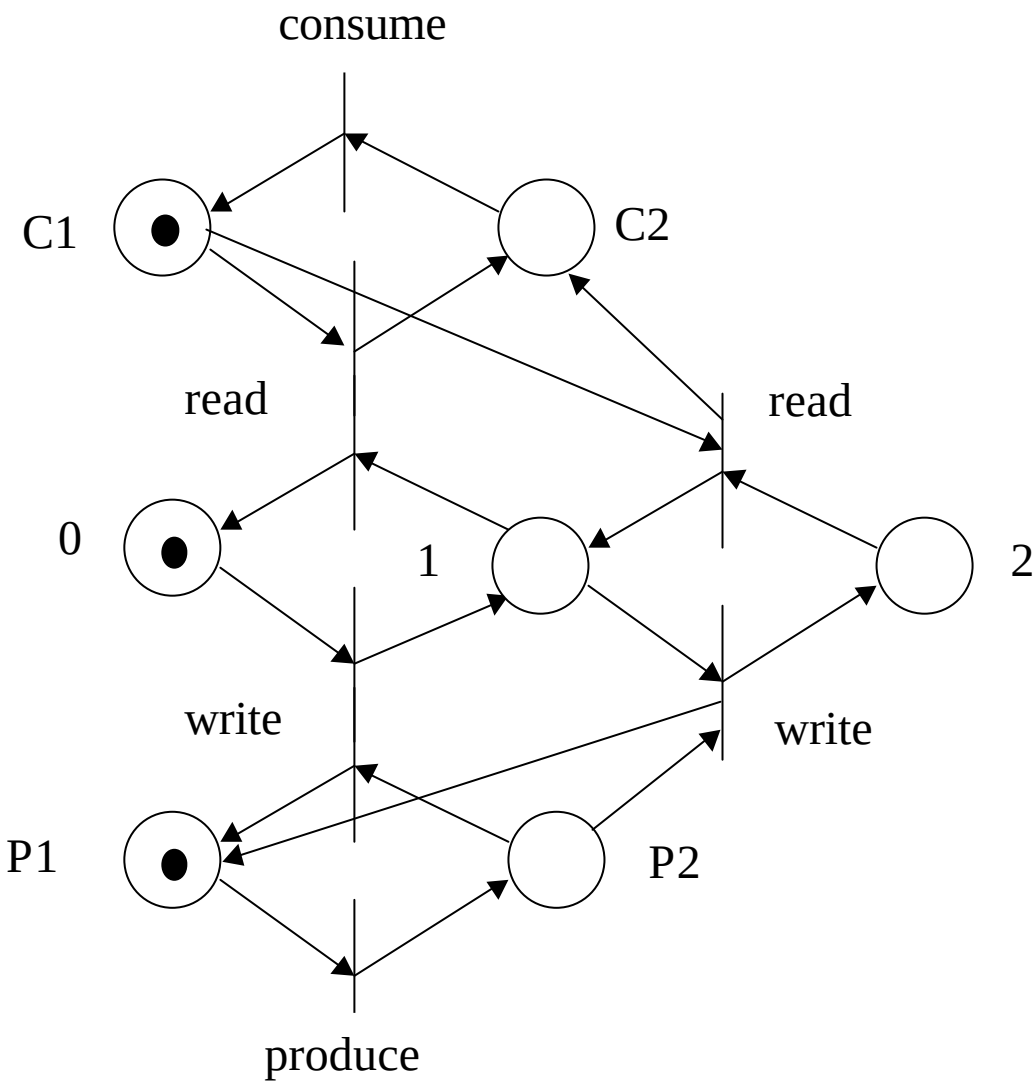
Reti di Petri (cont.)

Esempio:
produttore – consumatore (1)



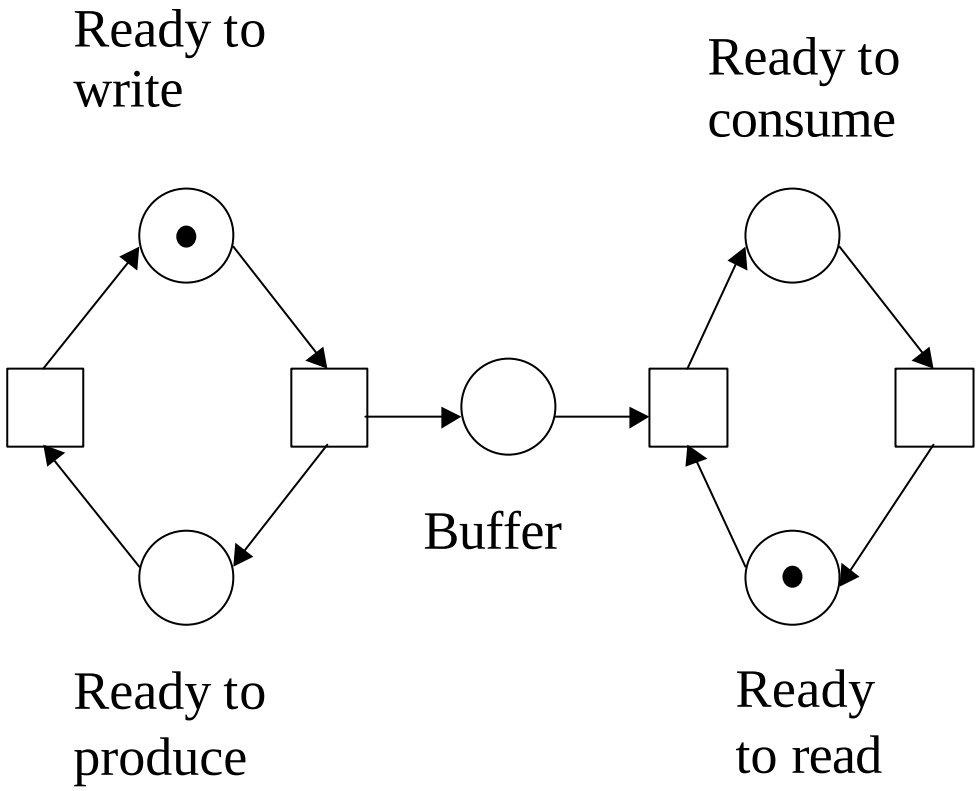
Reti di Petri (cont.)

Esempio:
produttore – consumatore (2)



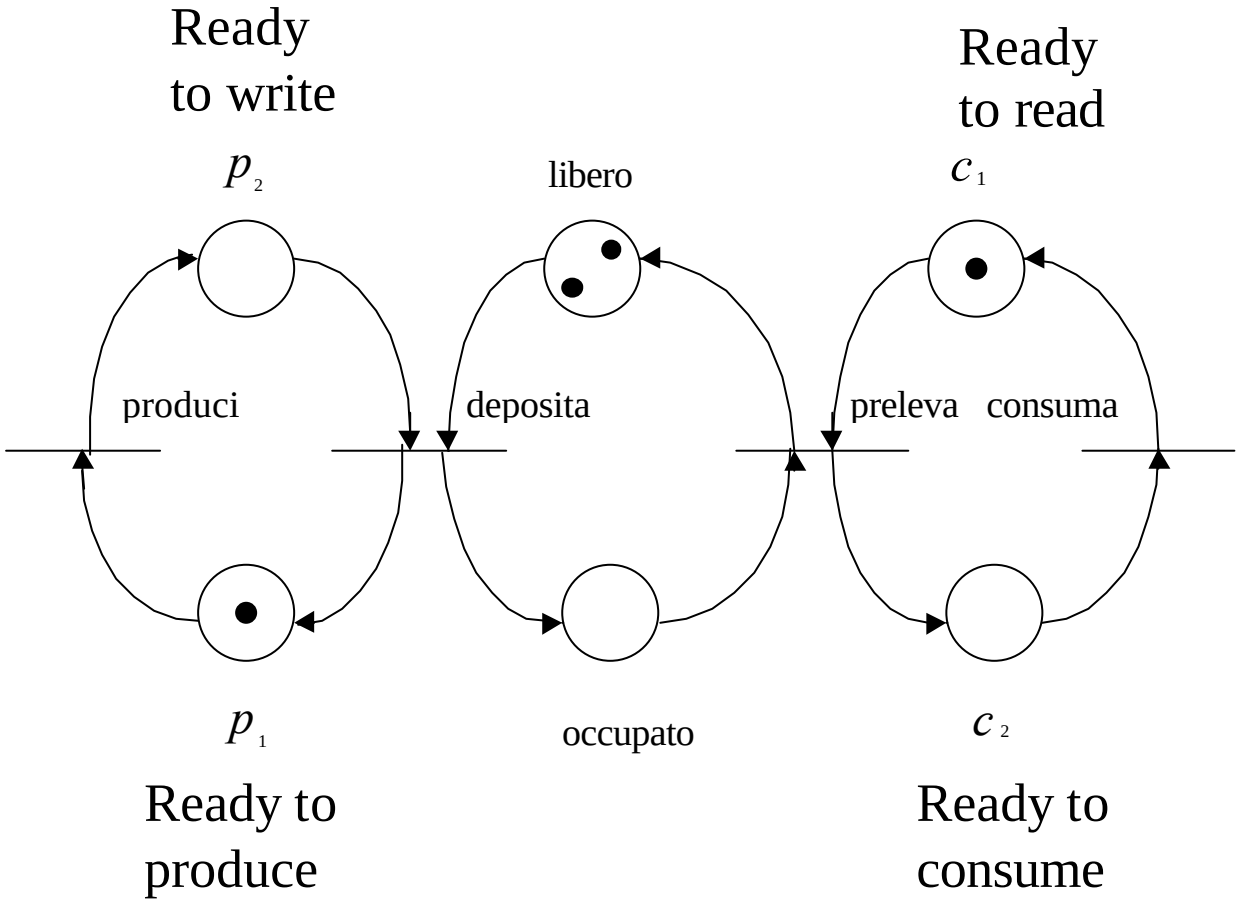
Reti di Petri (cont.)

Buffer illimitato



Reti di Petri (cont.)

Buffer finito

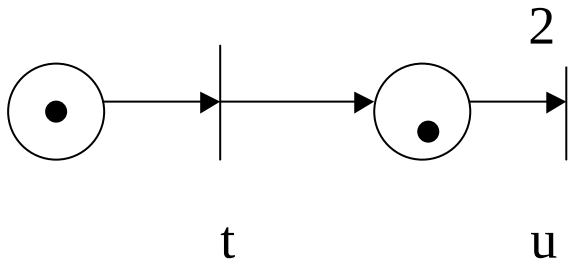


Reti di Petri (cont.)

Date due transizioni t e u, si definiscono

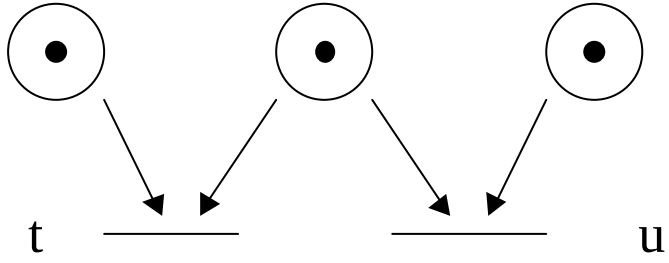
♦ Sequenza:

$M[t\rangle \wedge \neg M[u\rangle \wedge M[tu\rangle$



♦ Conflitto:

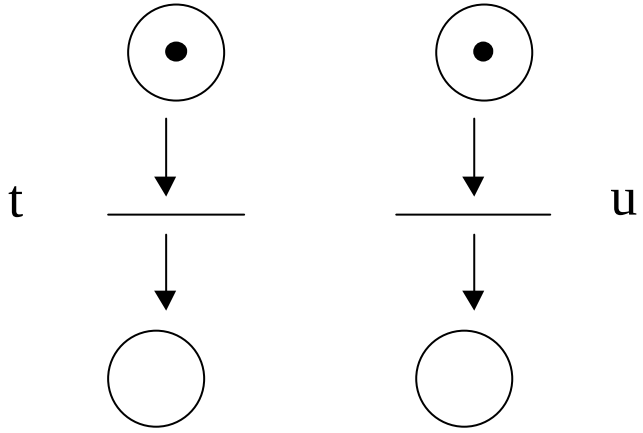
$M[t\rangle \wedge M[u\rangle \wedge \exists p \in Pre(t) \cap Pre(u) \bullet M(p) < W(\langle p, t \rangle) + W(\langle p, u \rangle)$



Reti di Petri (cont.)

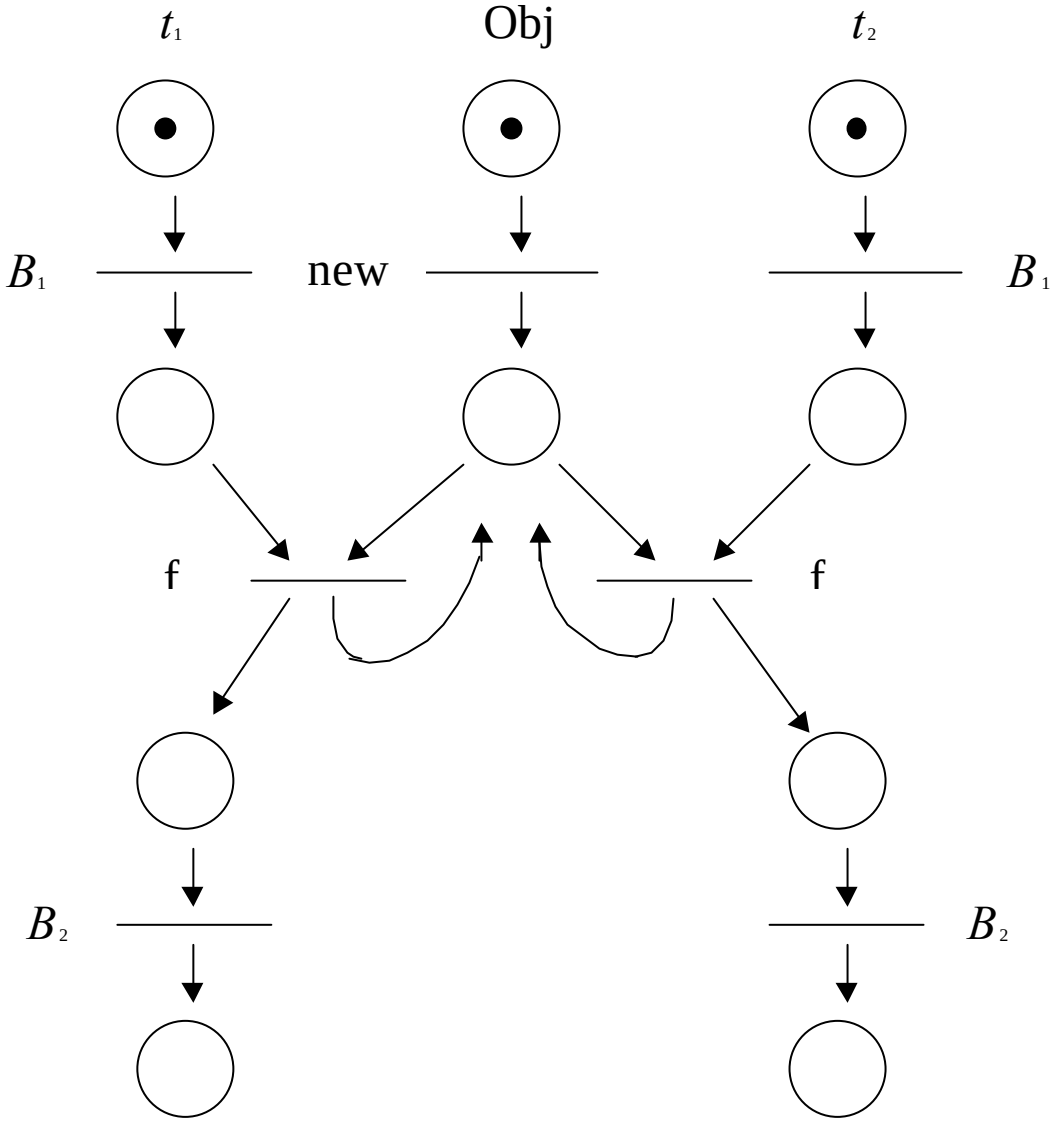
♦ Concorrenza:

$M[t\rangle \wedge M[u\rangle \wedge \forall\ p \in Pre(t) \cap Pre(u) \bullet M(p) \geq W(\langle p, t \rangle) + W(\langle p, u \rangle)$



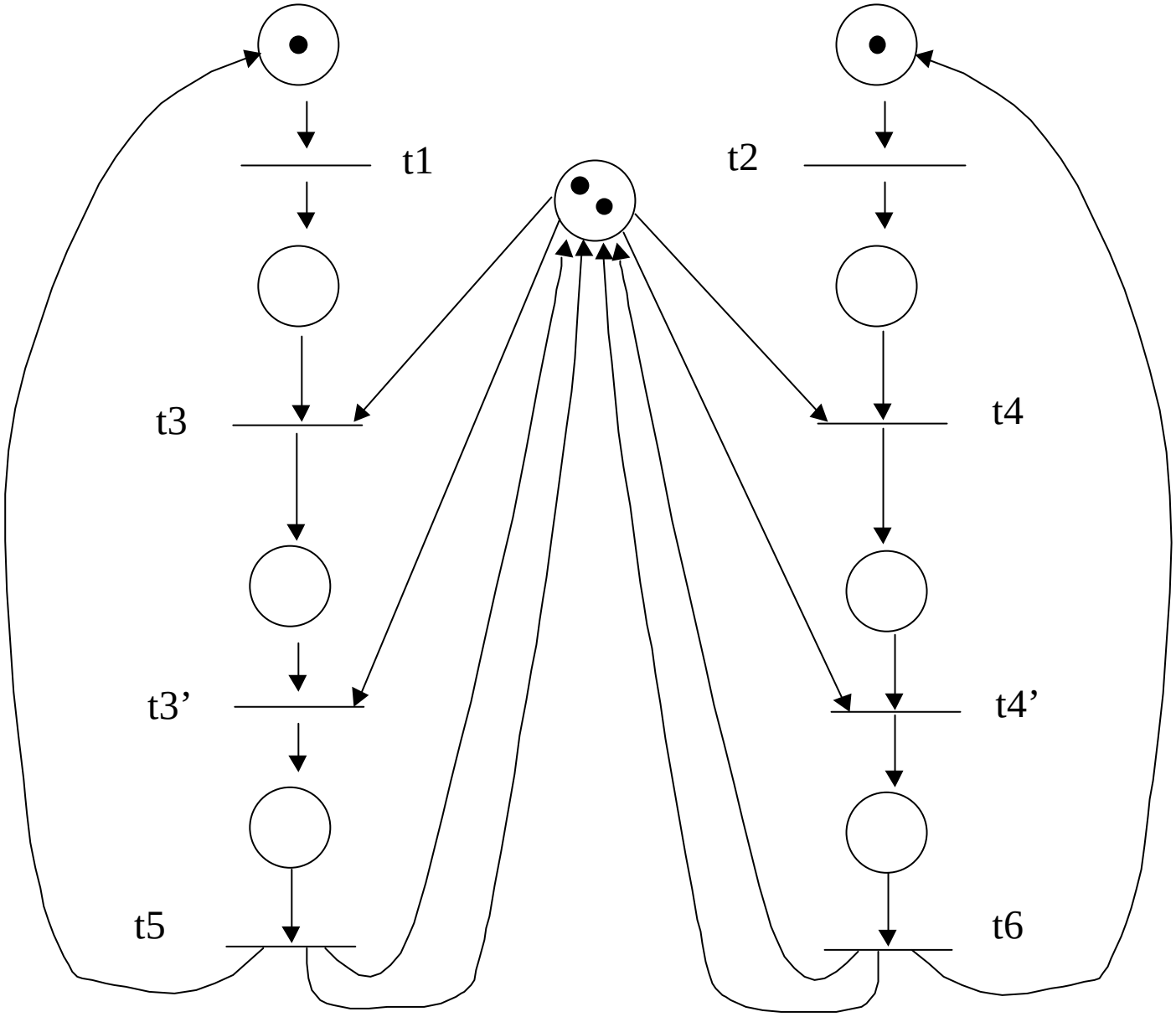
Reti di Petri (cont.)

- Modellizzazione di metodi sincronizzati (ad es. in Java)



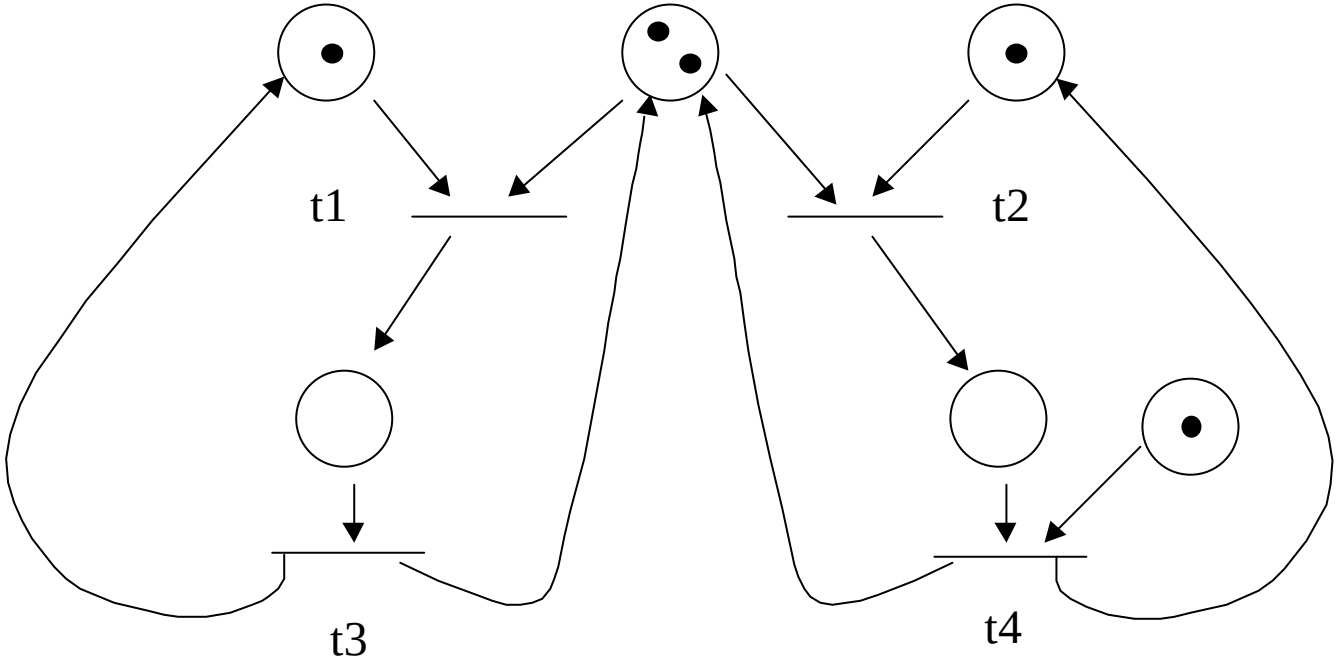
Reti di Petri (cont.)

- Deadlock



Reti di Petri (cont.)

♦ Starvation



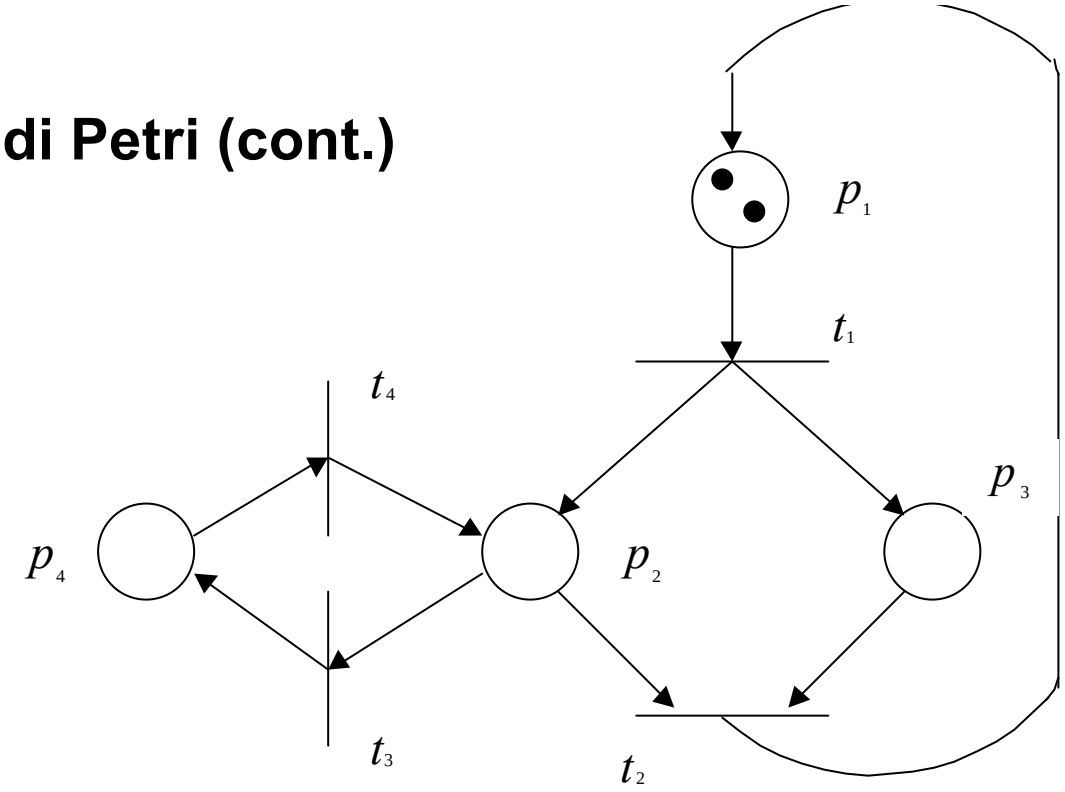
Reti di Petri (cont.)

Verifica delle proprietà: Analisi di raggiungibilità

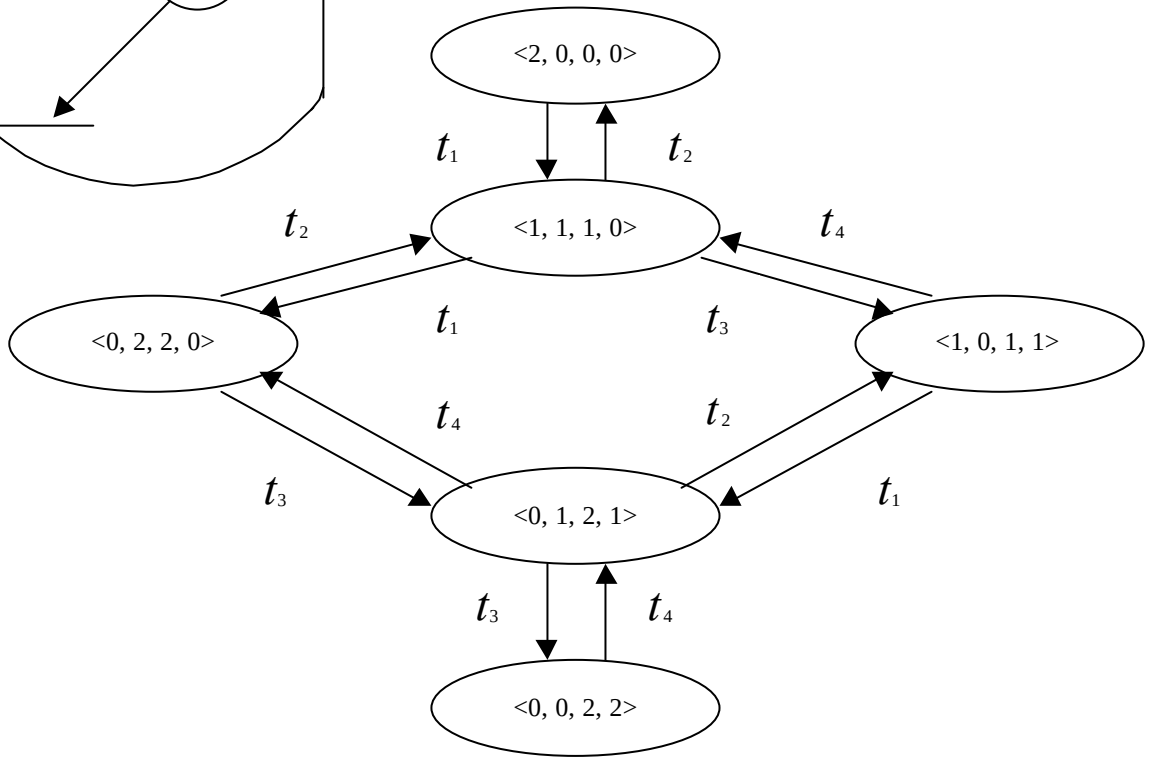
Consiste nel trovare le marcature raggiungibili a partire da una marcatura iniziale: problema decidibile ma di complessità esponenziale

Raggiungibilità:
M' è raggiungibile in un passo se $\exists\ t \bullet M[t > M'$.
M' è raggiungibile se $\exists\ t_1 \ldots t_k \bullet M[t_1 \ldots t_k > M'$.

Reti di Petri (cont.)

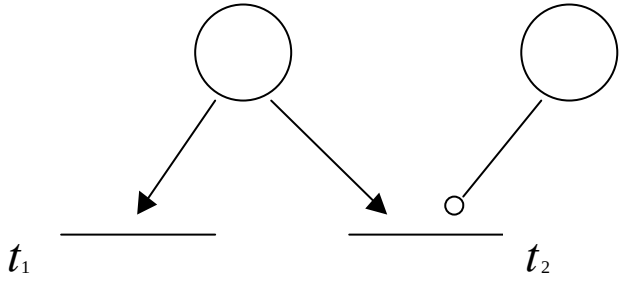


Grafo di raggiungibilità: nessun deadlock



Reti di Petri: estensioni

Archi inibitori: abilitano la transizione solo in assenza di token nel posto da cui provengono



Reti di Petri: estensioni (cont.)

Reti temporizzate (reti di Merlin&Farber)

- a ogni transizione è associato un intervallo $[t_{min}, t_{max}]$ che rappresenta il tempo minimo/massimo per lo scatto → cambia la condizione di abilitazione di una transizione

⇓

se una transizione è abilitata nel senso originale all'istante *t*, essa DEVE scattare fra l'istante $t + t_{min}$ e $t + t_{max}$, a meno che non venga disabilitata dallo scatto di un'altra transizione

- si presuppone l'esistenza di un orologio globale
- la rete è di tipo stocastico se per i tempi di scatto delle transizioni si forniscono le distribuzioni di probabilità

Reti di Petri: estensioni (cont.)