

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)

Requirements engineering

The process of establishing requirements, i.e. the SERVICES that the customer requires from a system and the CONSTRAINTS under which it operates and is developed

What is a requirement?

- Requirements may serve a dual function
 - May be the basis for a bid for a contract - therefore must be open to interpretation
 - May be the basis for the contract itself - therefore must be defined in detail
 - Both these statements may be called requirements

Types of requirement

- User requirements
 - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers
- System requirements
 - A structured document setting out detailed descriptions of the system services. Written as a contract between client and contractor
- Software specification
 - A detailed software description which can serve as a basis for a design or implementation. Written for developers

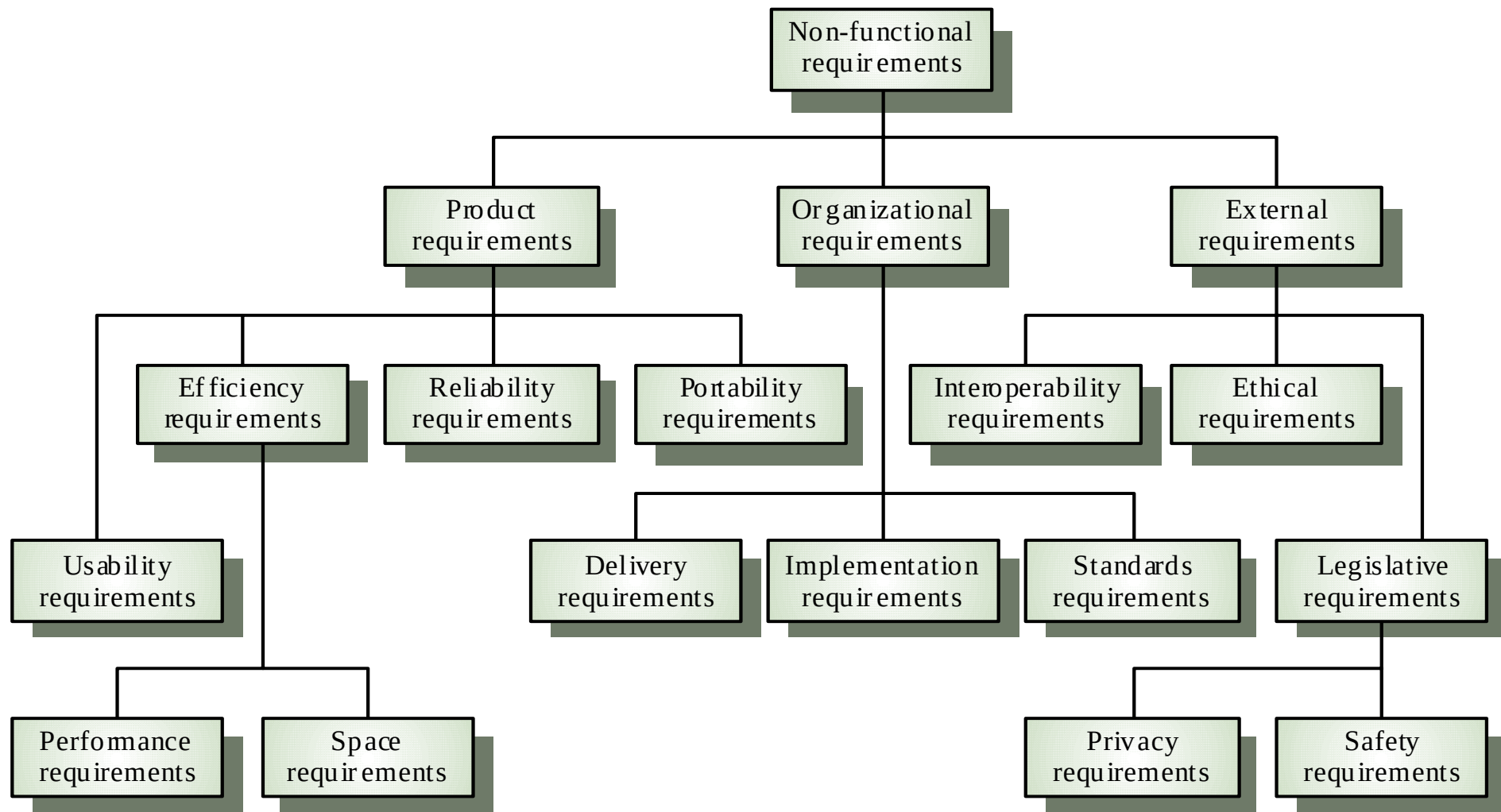
Functional and non-functional requirements

- Functional requirements
 - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- Non-functional requirements
 - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Domain requirements
 - Requirements that come from the application domain of the system and that reflect characteristics of that domain

Non-functional requirements

- Define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language or development method
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless

Non-functional requirement types



Domain requirements

- Derived from the application domain and describe system characteristics and features that reflect the domain
- May be new functional requirements, constraints on existing requirements or define specific computations
- If domain requirements are not satisfied, the system may be unworkable

Library system domain requirements

- There shall be a standard user interface to all databases which shall be based on the Z39.50 standard.
- Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.

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)

Requirements measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

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

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

Specifica dei requisiti

Tipologie di applicazioni (o di parti di esse)	Requisiti da specificare
Sequenziali (unico flusso di controllo)	Funzionalità offerte
Concorrenti (più flussi 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 risultati dipende dal tempo di esecuzione delle attività)	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
<i>Informali</i>	linguaggio naturale (che è impreciso, ambiguo e ridondante)	
<i>Formali</i> : 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
<i>Semiformali</i> : 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)

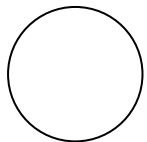
Diagramma di flusso dati (DFD)

- 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

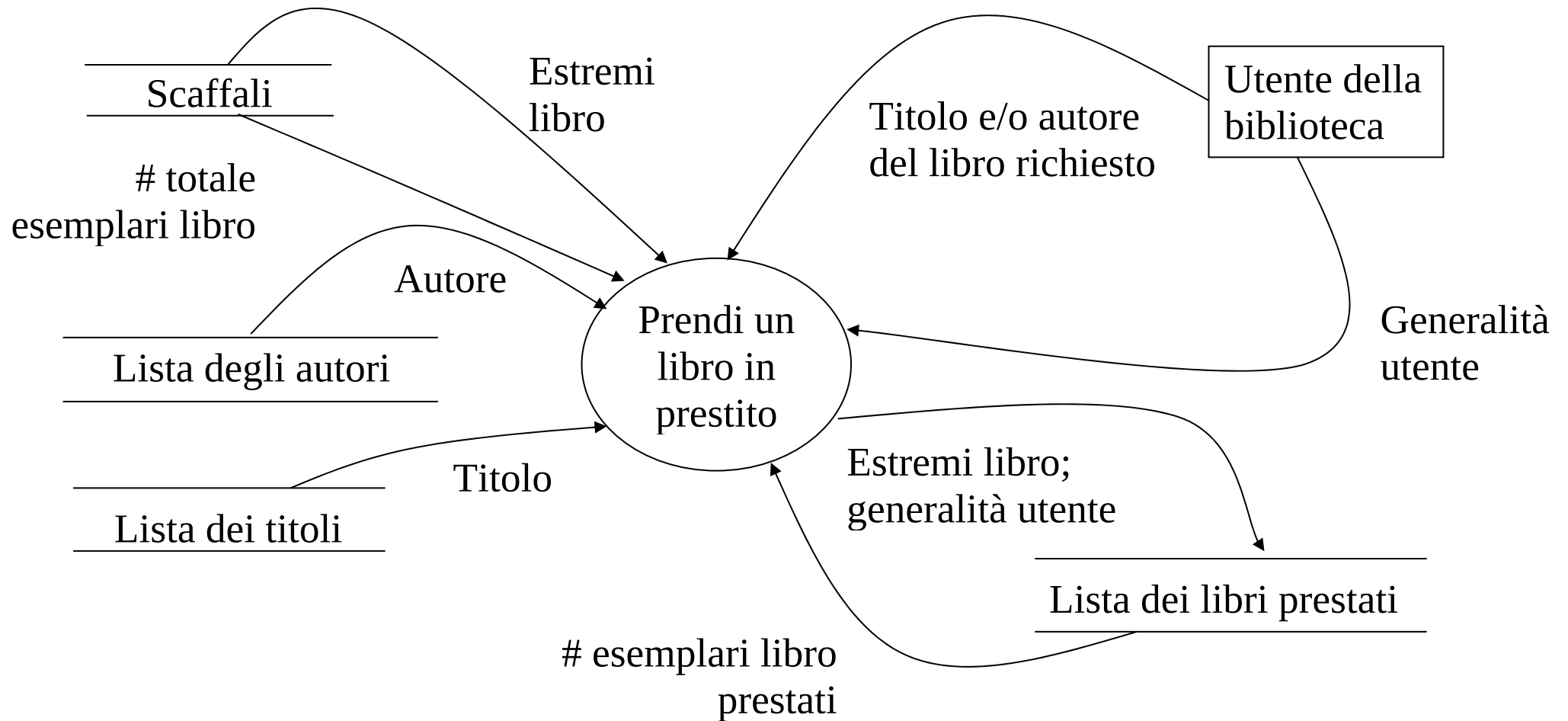


Diagramma di flusso dati (cont.)

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”

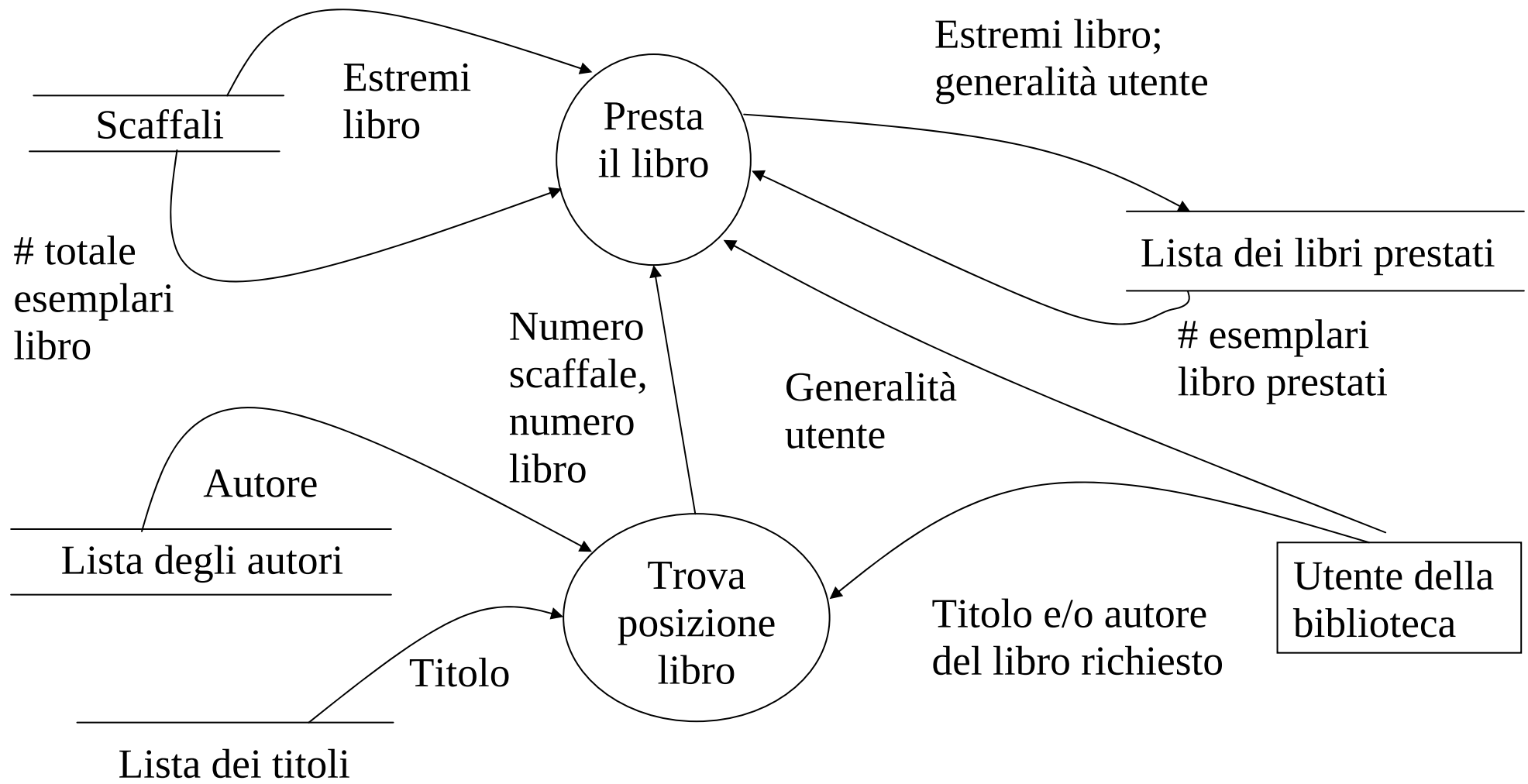


Diagramma di flusso dati (cont.)

Linee guida per la costruzione di un DFD:

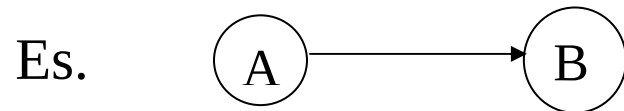
- ignorare inizializzazione, terminazione e casi di errore; concentrarsi sulle condizioni stabili
- ignorare flusso di controllo e sincronizzazione
- individuare innanzi tutto ingressi e uscite esterni al (sotto) sistema che si sta descrivendo
- usare nomi significativi
- percorrere i flussi informativi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

Pregio dei DFD: sono facili da leggere

Limiti dei DFD:

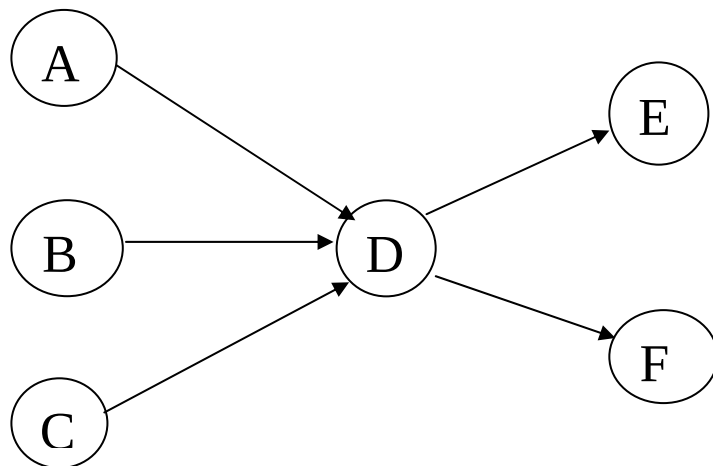
- non consentono di esprimere il controllo di flusso né le sincronizzazioni
- sono ambigui

Diagramma di flusso dati: ambiguità



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)

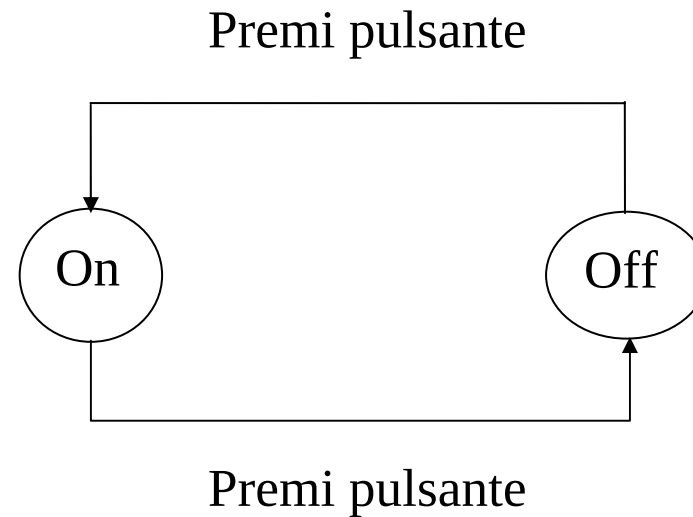
S: insieme finito di stati

I: insieme finito degli ingressi

δ : funzione di transizione

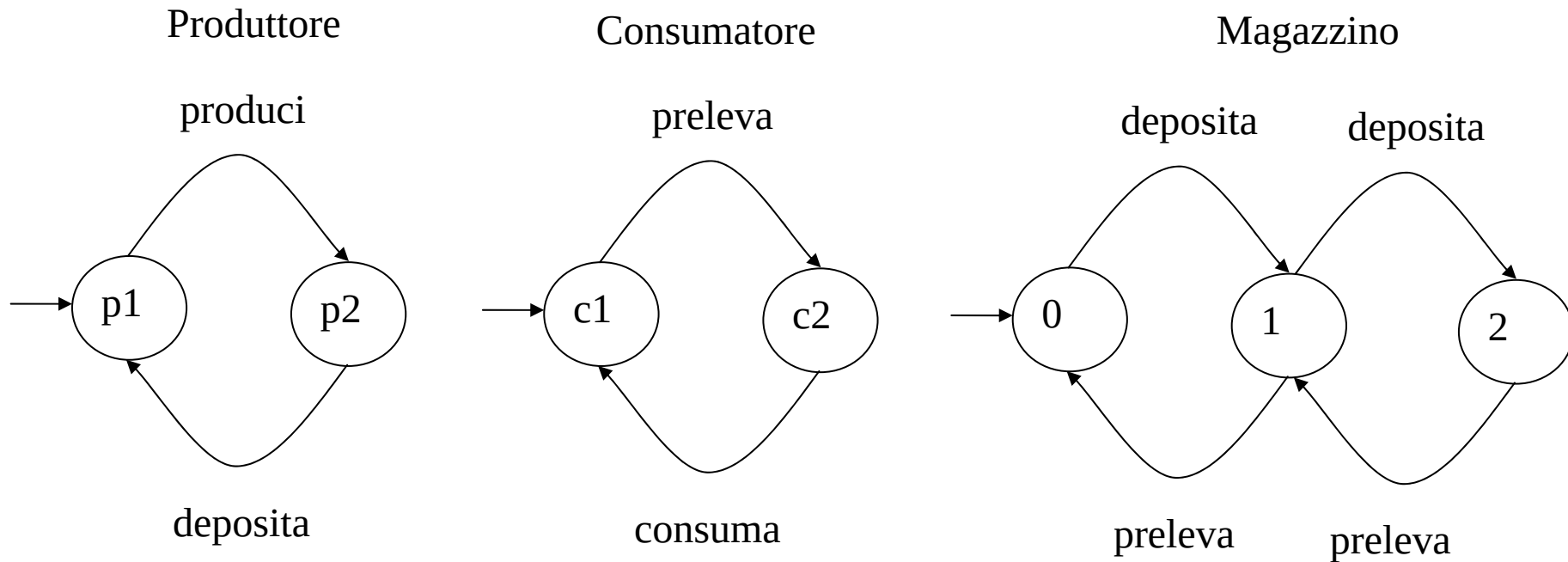
Classi di FSM

- Deterministiche/nondeterministiche
- Riconoscitori
- Traduttori



Macchine a stati finiti (cont.)

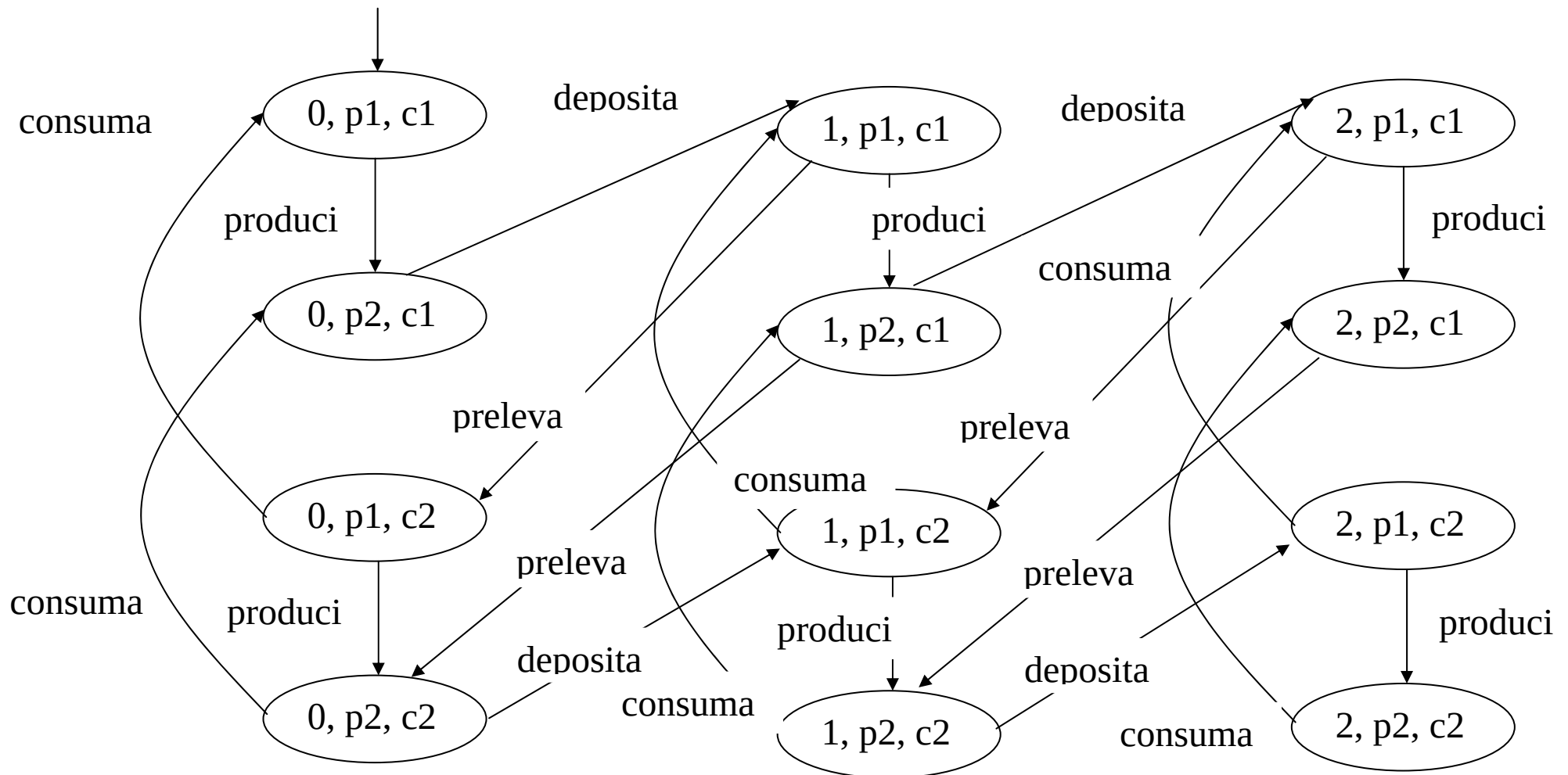
Esempio: produttore-consumatore



Limitazione

Esplosione degli stati quando si rappresentano sistemi concorrenti: date n FSM con $k_1, k_2, \dots, k_n$ stati, la loro composizione è una FSM con $k_1 * k_2 * \dots * k_n$ stati (mentre vorremmo che fossero $k_1 + k_2 + \dots + k_n$)

Macchine a stati finiti (cont.)

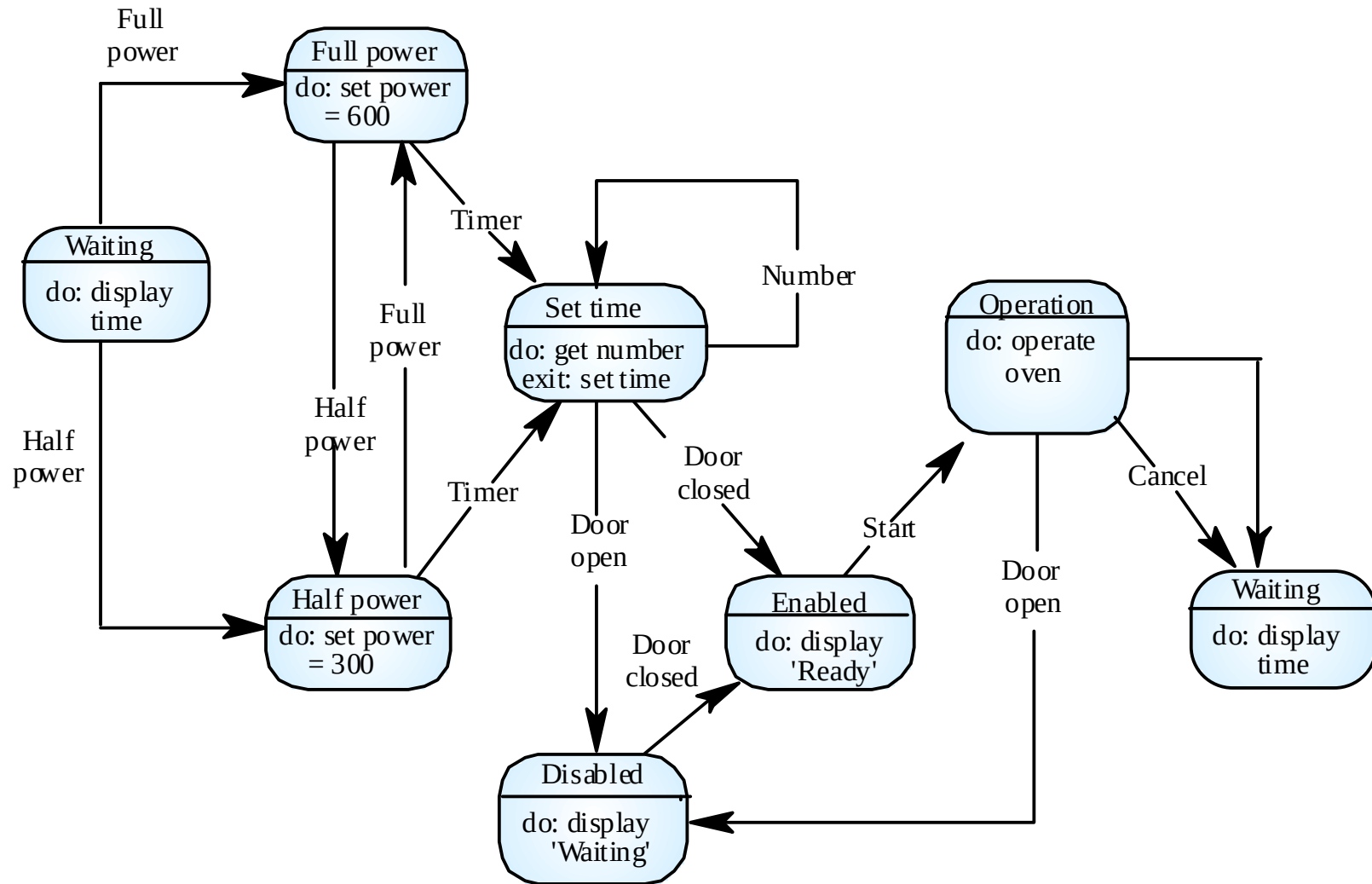


Macchine a stati finiti (cont.)

Soluzione della limitazione delle FSM:

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

Statechart



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

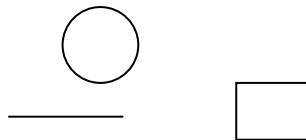
$$N = (P, T, F)$$

con

P: insieme finito di posti (places)

T: insieme finito di transizioni

F: relazione di flusso



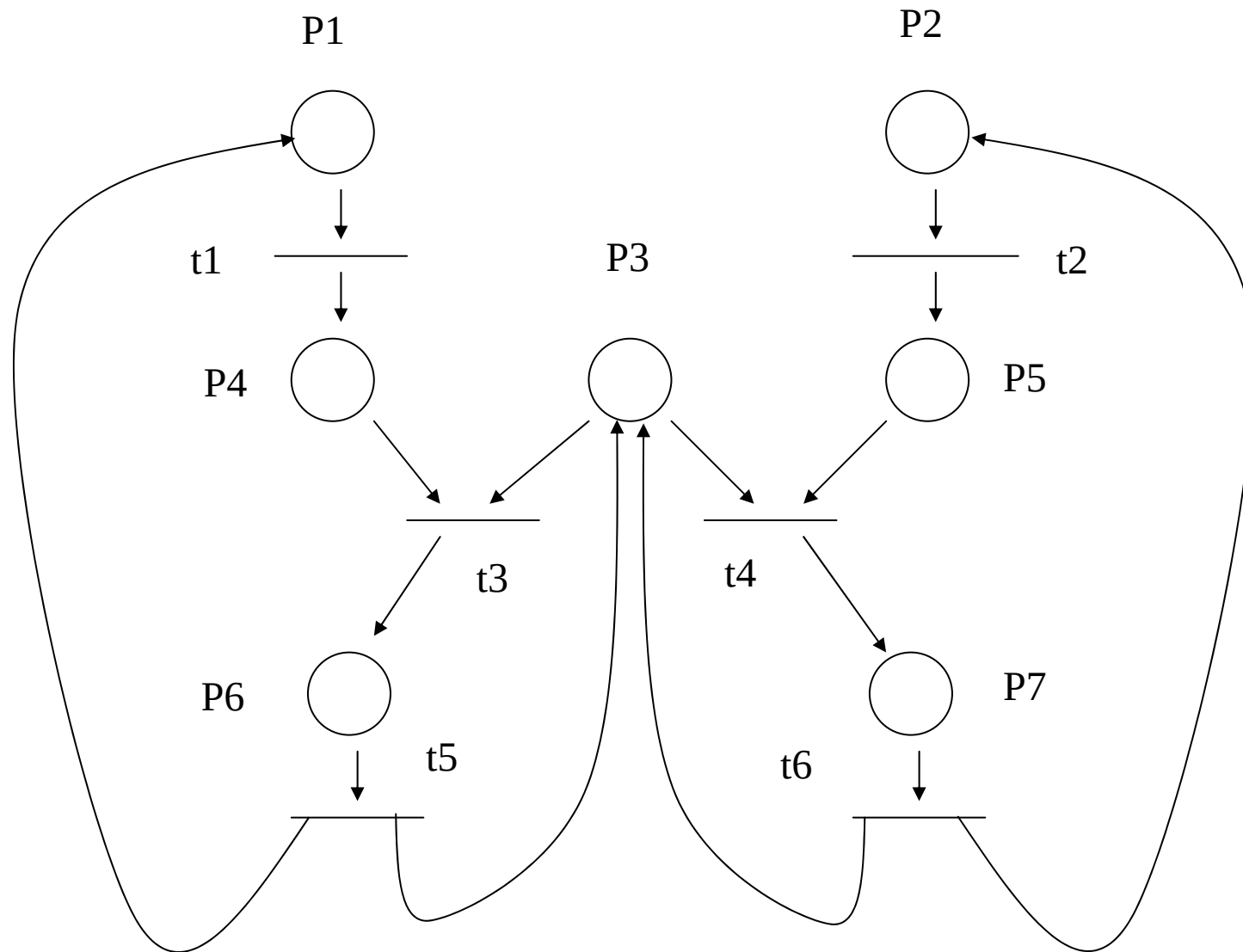
Vincoli:

$$(1) P \cap T = \emptyset$$

$$(2) P \cup T \neq \emptyset$$

$$(3) F \subseteq (P \times T) \cup (T \times P)$$

Reti di Petri (cont.)



Reti di Petri (cont.)

$$\text{Pre}(y) = \{ x \in P \cup T \mid \langle x, y \rangle \in F \}$$

$$\text{Post}(x) = \{ y \in P \cup T \mid \langle x, y \rangle \in F \}$$

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

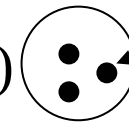
W: peso

M_0 : marcatura iniziale

Vincoli:

(4) $W: F \rightarrow N - \{0\}$, valore di default di W è 1 $\xrightarrow{4}$

(5) $M_0: P \rightarrow N$ (un funzione $M: P \rightarrow N$ è chiamata *marcatura*)



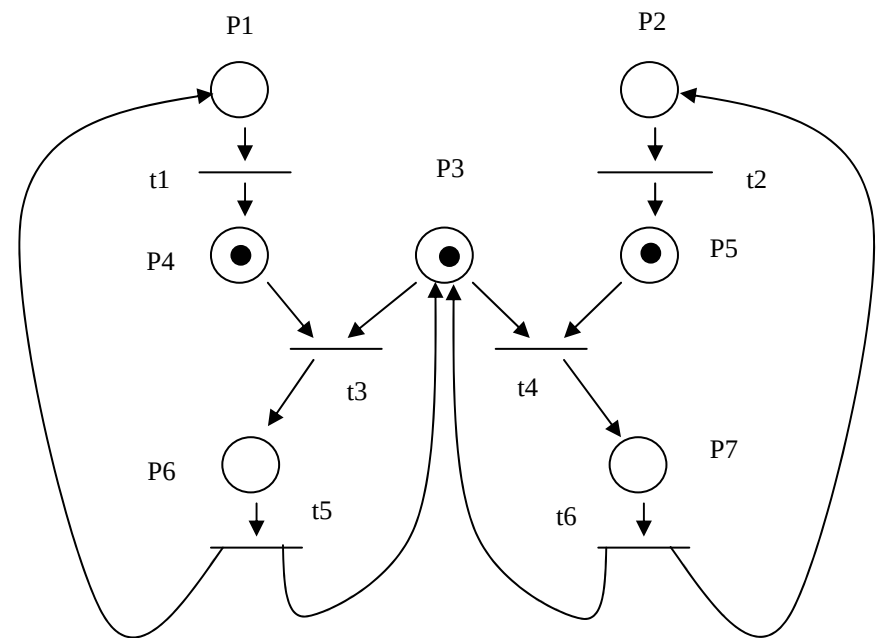
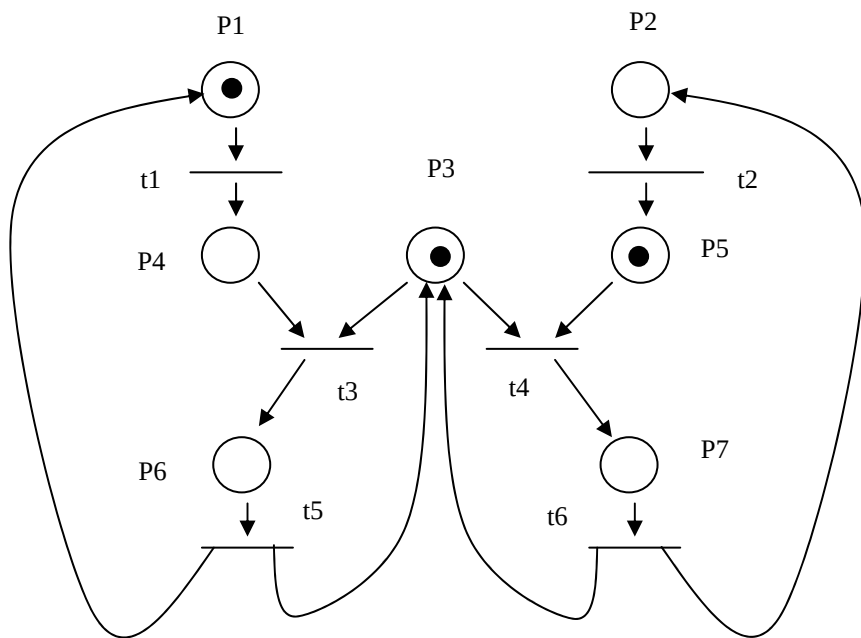
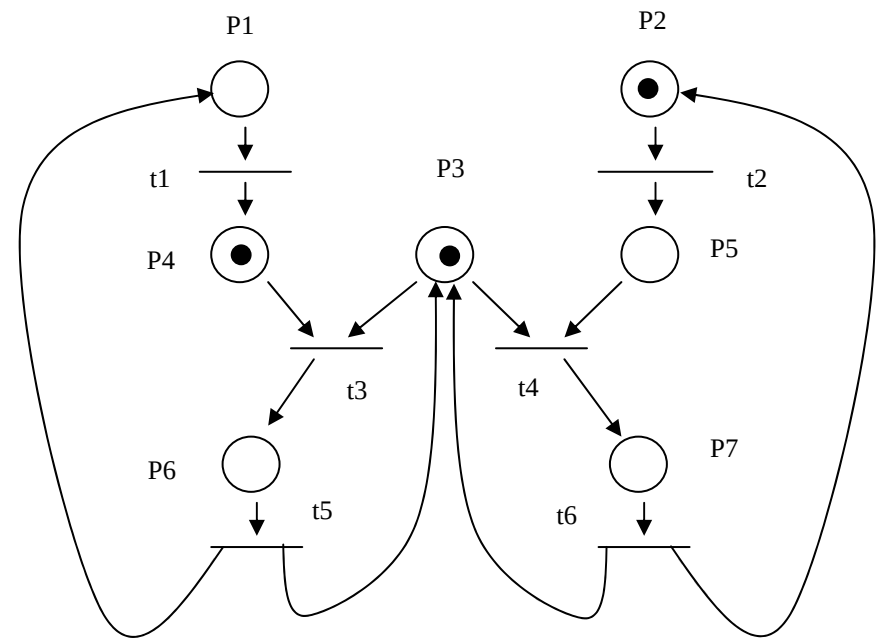
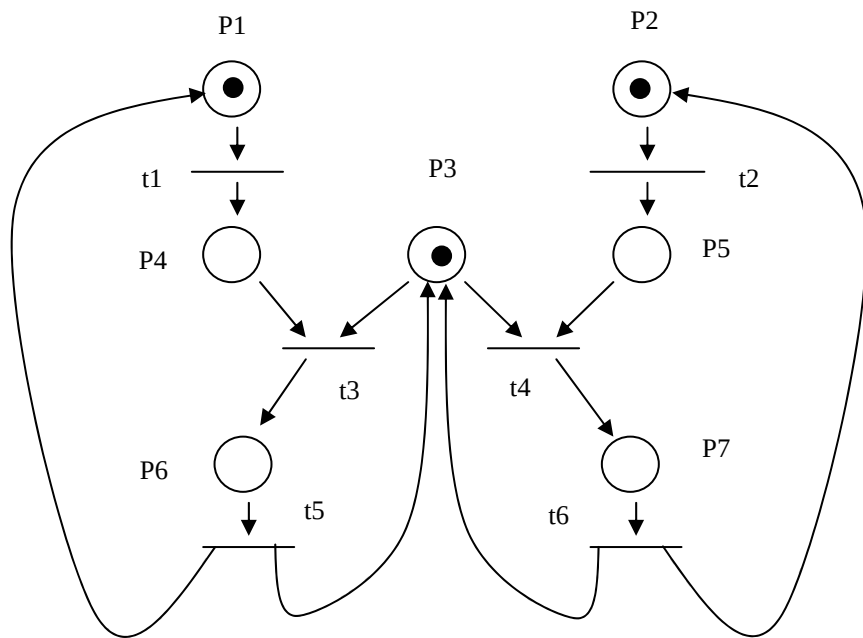
token

- 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*: $M_0 \Rightarrow M_1 \Rightarrow M_2 \Rightarrow \dots$

Reti di Petri (cont.)

Semantica: descrizione informale (caso $W=1$)

- ◆ una transizione t è *abilitata* se c'è (almeno) un token in ogni $p \in \text{Pre}(t)$
- ◆ una transizione t abilitata può scattare, dando luogo a una nuova marcatura dove:
 - un token è cancellato da ogni $p \in \text{Pre}(t)$
 - un token è aggiunto a ogni $p \in \text{Post}(t)$
- ◆ se esistono più transizioni abilitate, la scelta della transizione da eseguire è nondeterministica



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 ($M[t > M']$):

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

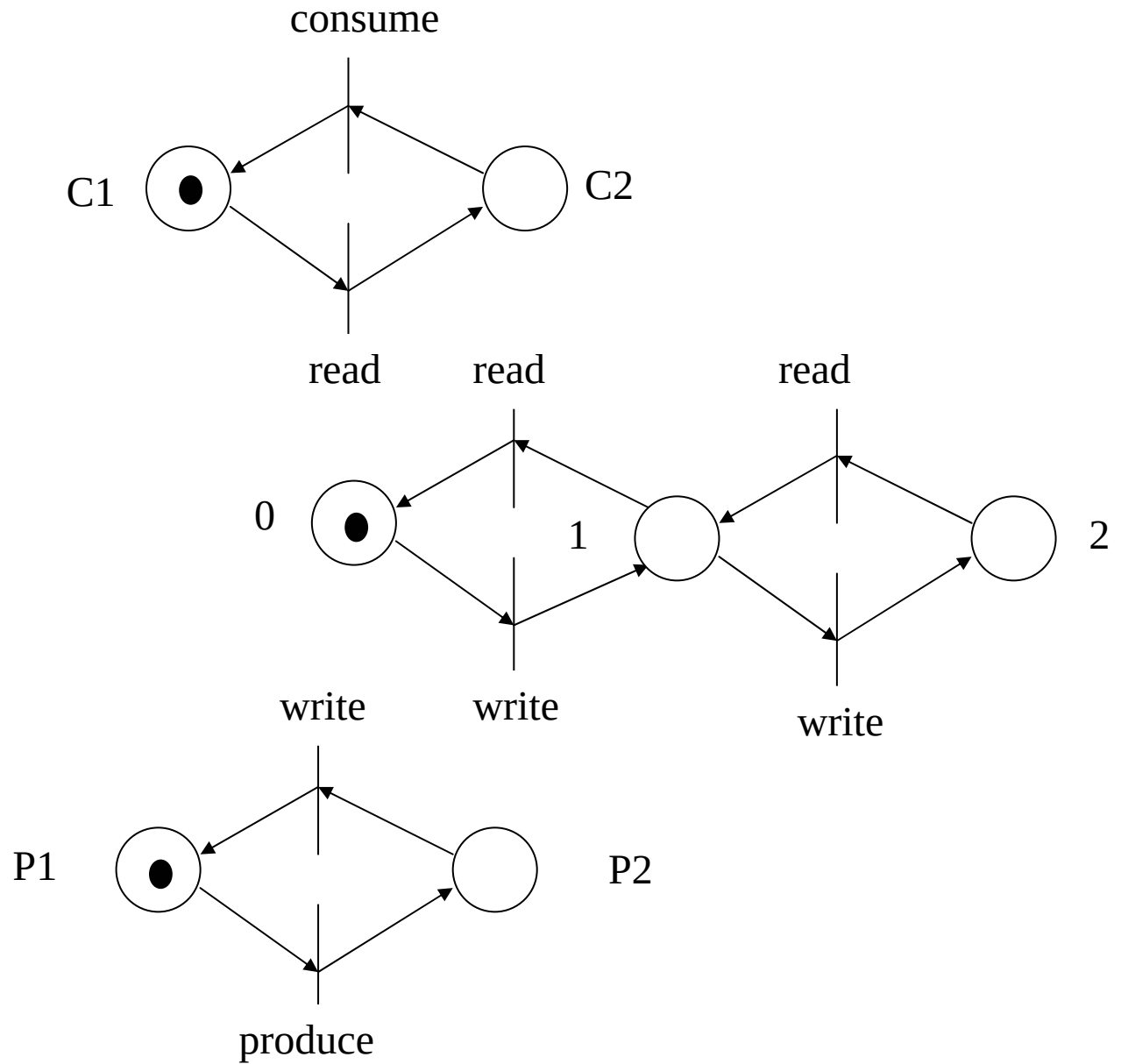
$$\forall p \in \text{Post}(t) - \text{Pre}(t) \bullet M'(p) = M(p) + W(\langle t, p \rangle)$$

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

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

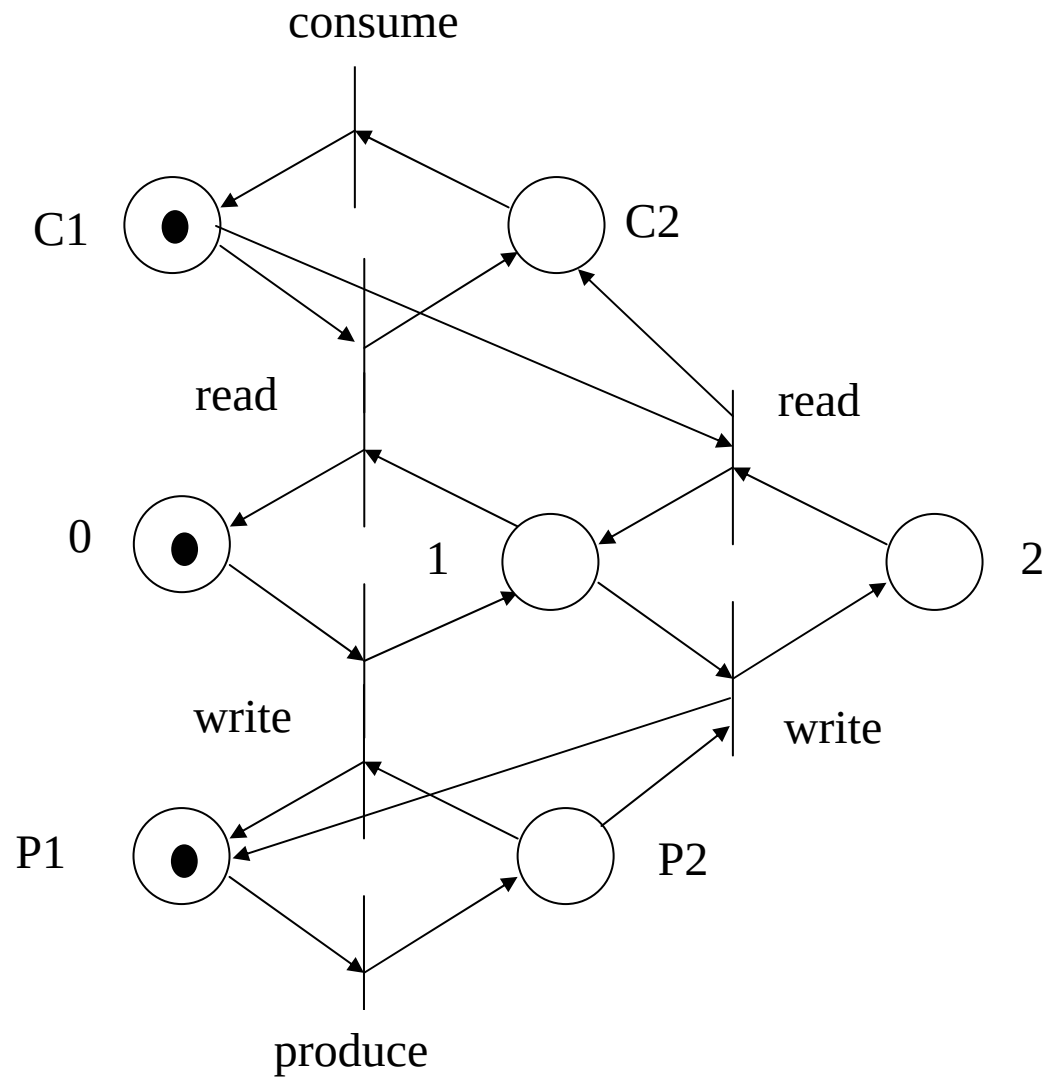
Reti di Petri (cont.)

Esempio:
produttore – consumatore (1)



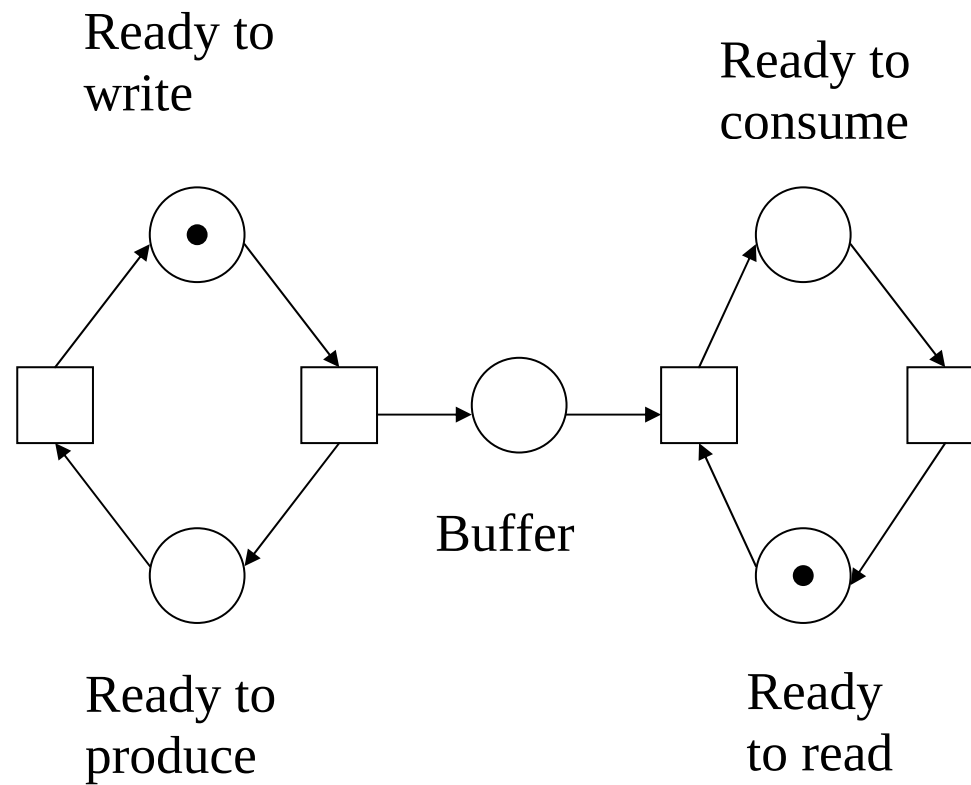
Reti di Petri (cont.)

Esempio:
produttore – consumatore (2)



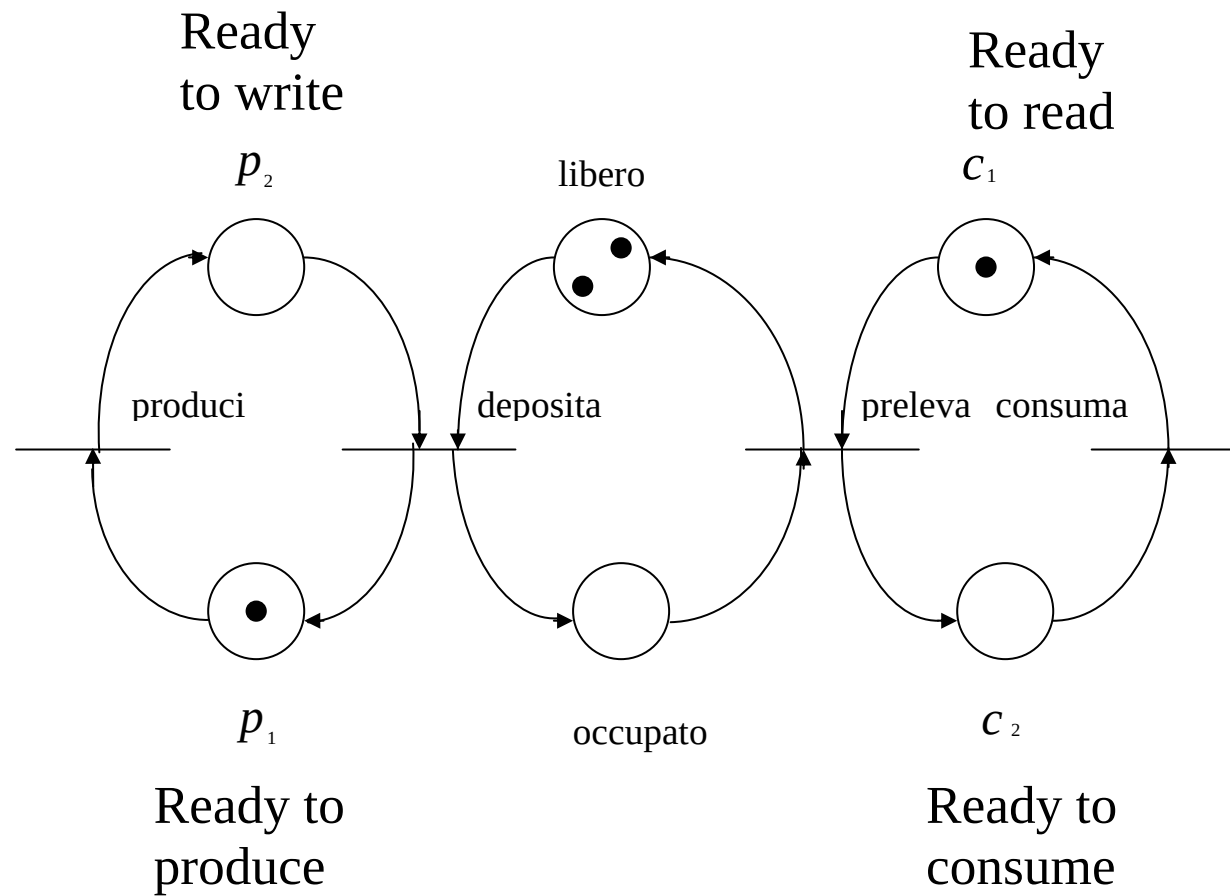
Reti di Petri (cont.)

Buffer illimitato



Reti di Petri (cont.)

Buffer finito

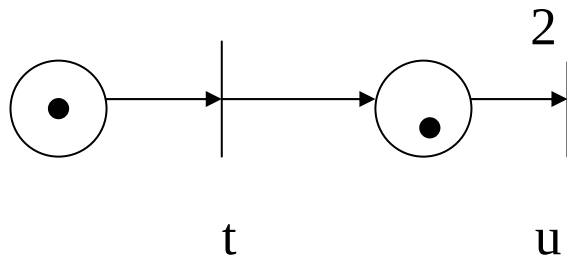


Reti di Petri (cont.)

Date due transizioni t e u , si definiscono

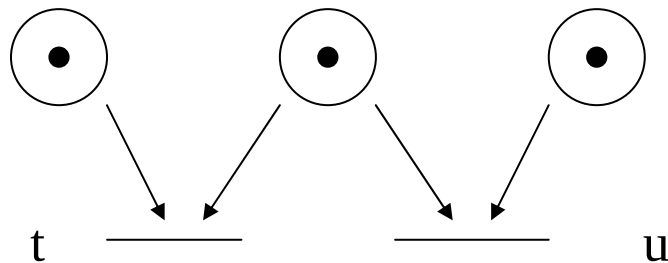
♦ Sequenza:

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



♦ Conflitto:

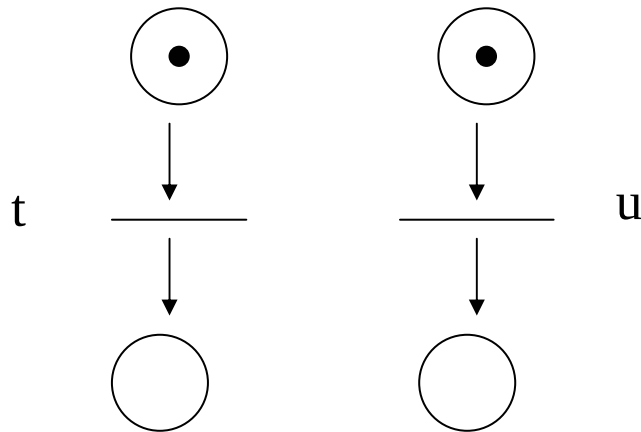
$$M[t] \wedge M[u] \wedge \exists p \in \text{Pre}(t) \cap \text{Pre}(u) \bullet M(p) < W(\langle p, t \rangle) + W(\langle p, u \rangle)$$



Reti di Petri (cont.)

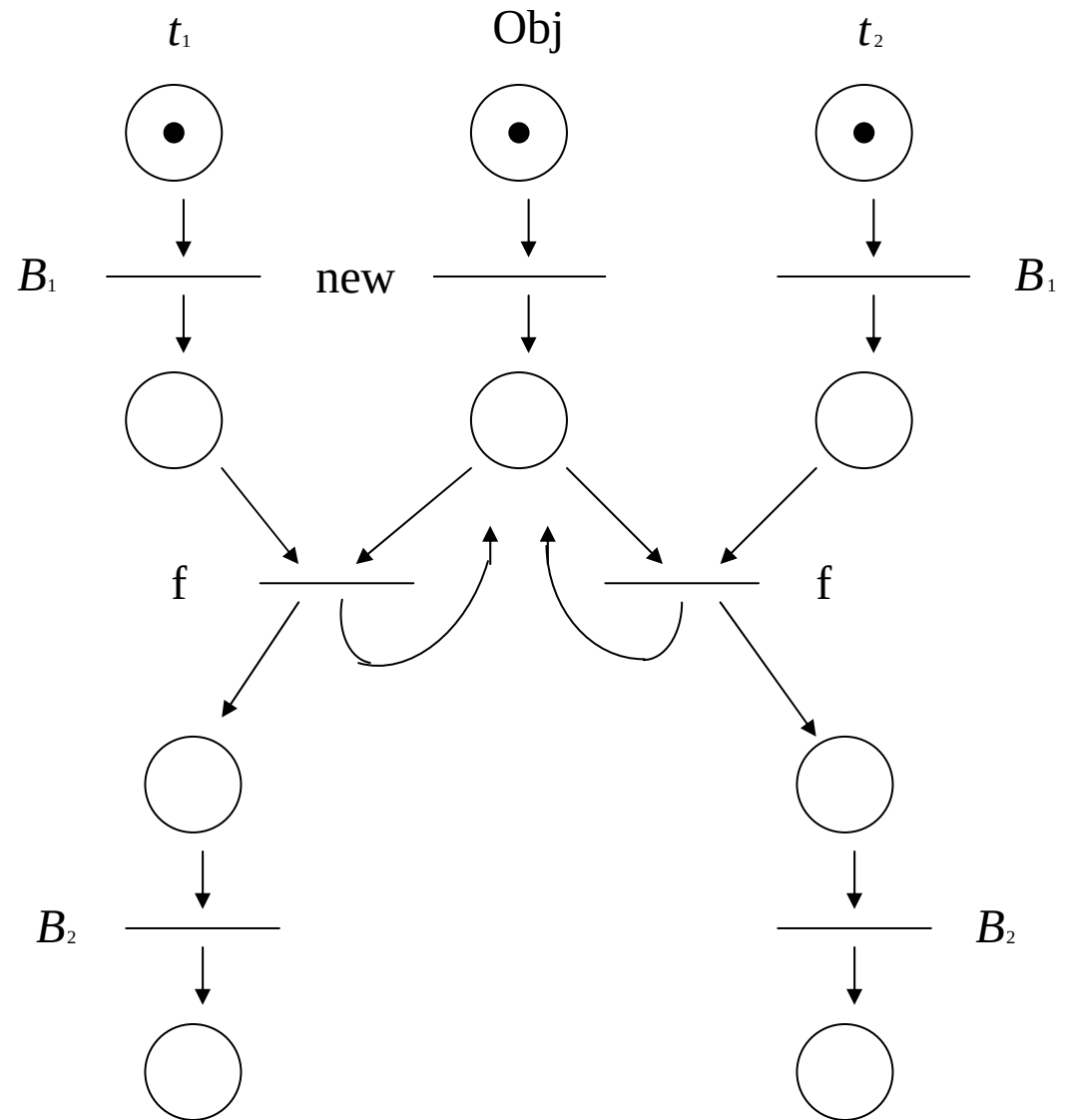
◆ Concorrenza:

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



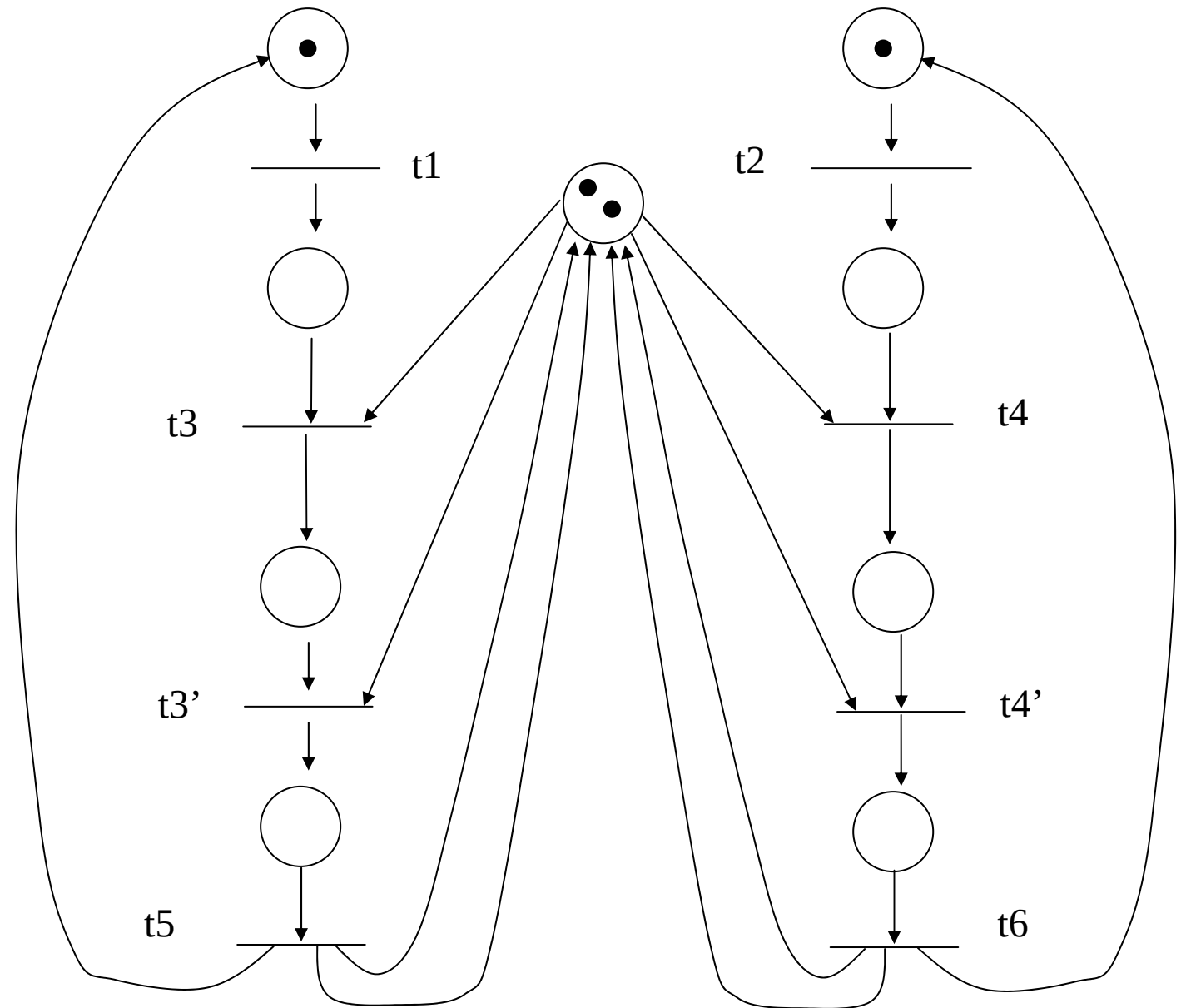
Reti di Petri (cont.)

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



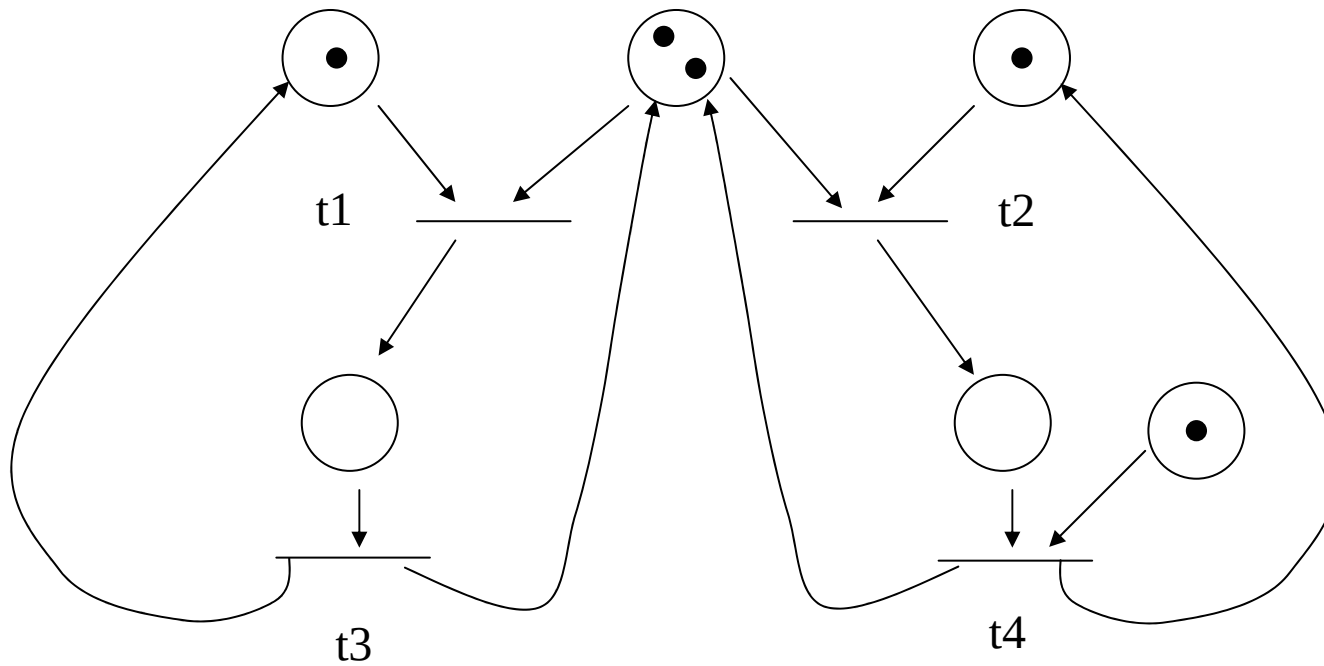
Reti di Petri (cont.)

◆ Deadlock



Reti di Petri (cont.)

◆ Starvation



Reti di Petri (cont.)

Verifica delle proprietà: Analisi di raggiungibilità

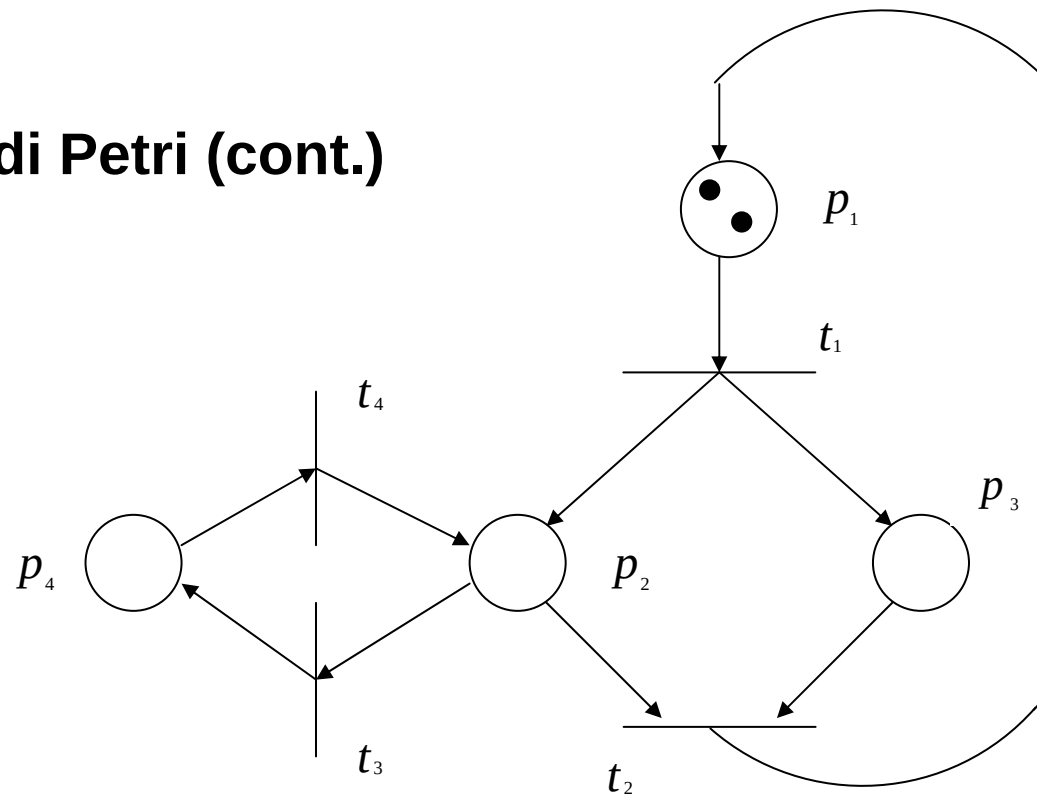
Consiste nel trovare le marcature raggiungibili a partire da una marcatura iniziale: problema decidibile ma di complessità esponenziale

Raggiungibilità:

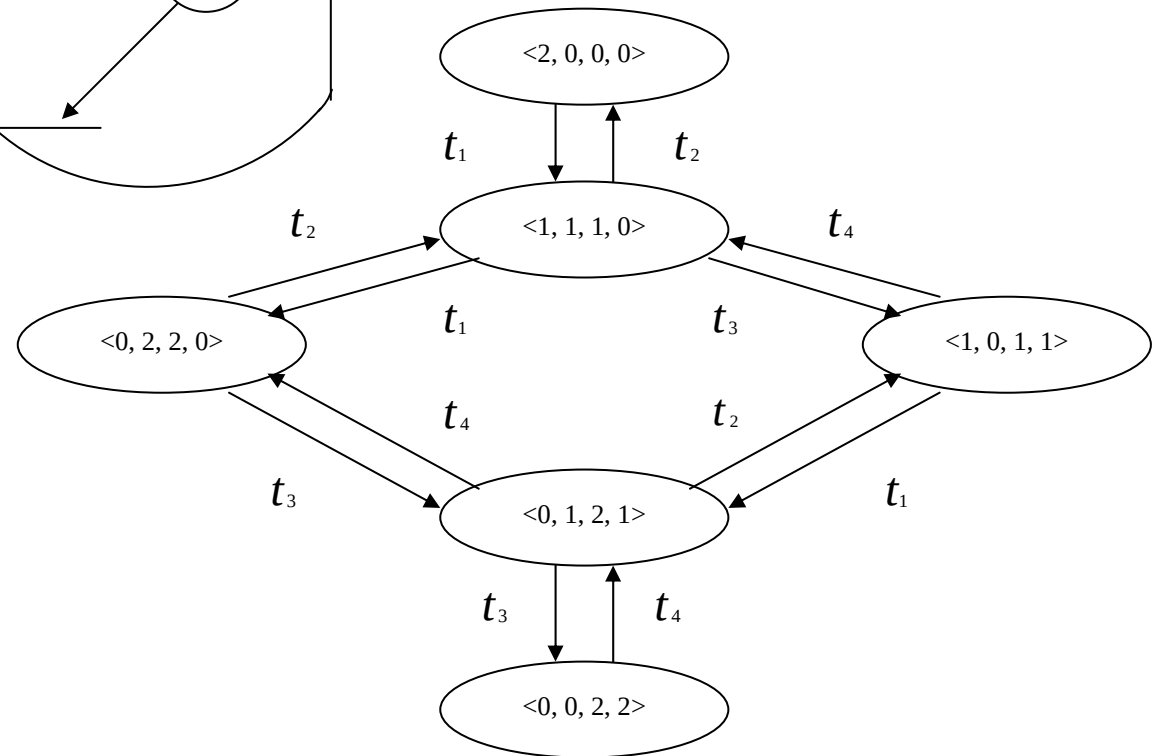
M' è raggiungibile in un passo se $\exists t \bullet M[t > M'$.

M' è raggiungibile se $\exists t_1 \dots t_k \bullet M[t_1 \dots t_k > M'$.

Reti di Petri (cont.)

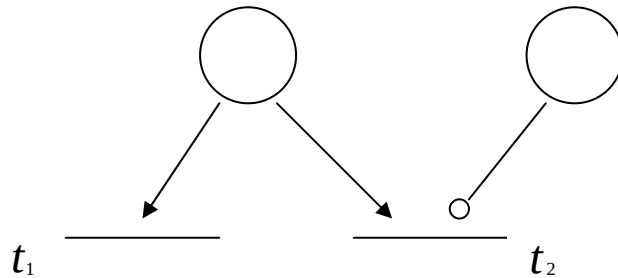


Grafo di
raggiungibilità:
nessun deadlock



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 $\rightarrow$ 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.)

