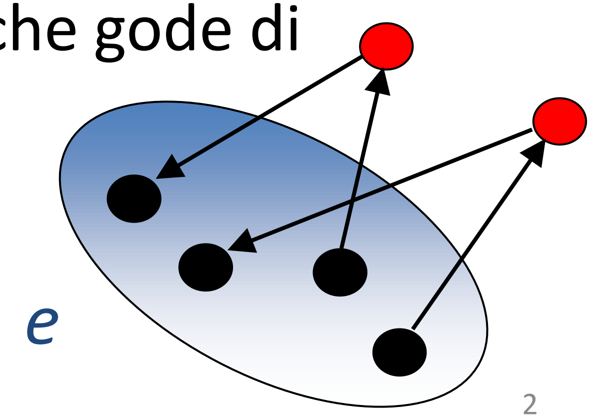
Algoritmi e Strutture Dati

Elaborato a.a. 2012-2013

Problema

- Input: un grafo orientato $\Gamma = (A, R)$
- Output: un insieme E_P di insiemi *e* di nodi appartenenti ad A, dove ciascun *e* gode delle seguenti proprietà
  - Non esistono archi (appartenenti a R) che uniscano alcuna coppia di nodi contenuti in *e*
  - Per ogni arco $(n_{ne}, n_e) \in R$, dove $n_{ne} \notin e$, $n_e \in e$, esiste un arco $(n'_e, n_{ne}) \in R$, dove $n'_e \in e$ (n'_e potrebbe anche coincidere con n_e)
  - Non esiste alcun superinsieme di *e* che gode di entrambe le proprietà precedenti



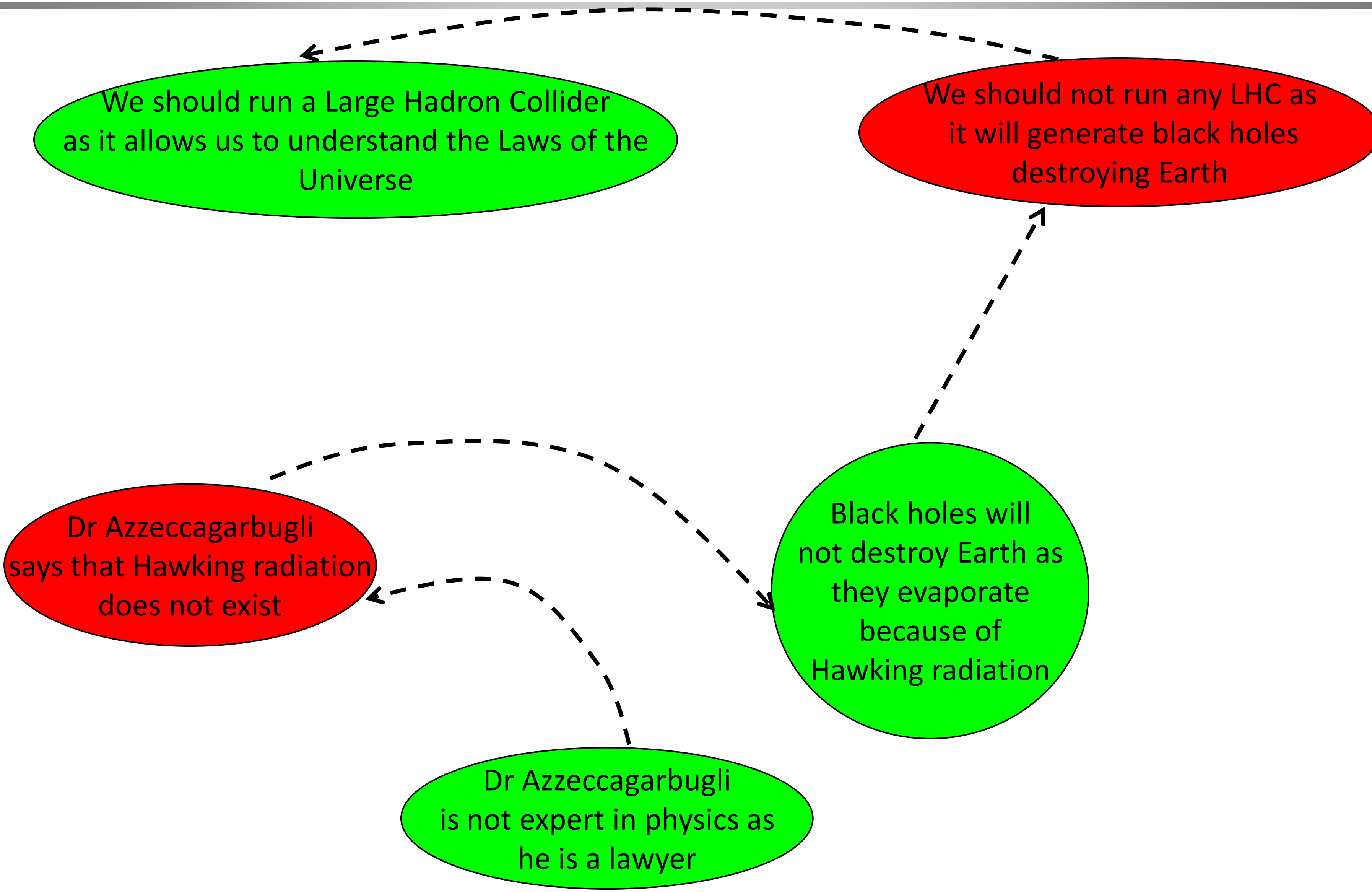
Dominio motivante: Argomentazione

- A: argomenti
- R: attacchi fra argomenti

- In tale dominio, E_P è l'insieme di tutte le "preferred extension" (PE), cioè ogni *e* è una PE

- Un sottografo si dice attaccato se subisce attacchi da parte di nodi esterni al sottografo stesso

An informal example



Algoritmo risolvente

- **Pref**(Γ,C)   // Γ = (A,R)
- //C ⊆A, alla prima invocazione C=A

- (*e*,I) ← **Grounded**(Γ,C)
- E_P ← {*e*}
- if I = ∅ return E_P

- Γ ← Γ↓I
- //Γ↓I è il sottografo di Γ contenente solo i nodi in I e //i relativi archi (non sospesi)

- S ← **SCCSSEQ**(Γ)
- //S è una sequenza di SCC; comprende tutti gli SCC contenuti in Γ secondo un ordine tale per cui S[0] non subisce attacchi e, per ogni i∈[2..length[S]], S[i] subisce eventuali attacchi solo da parte degli SCC che lo precedono in S (indicati cumulativamente come S[1..i-1])

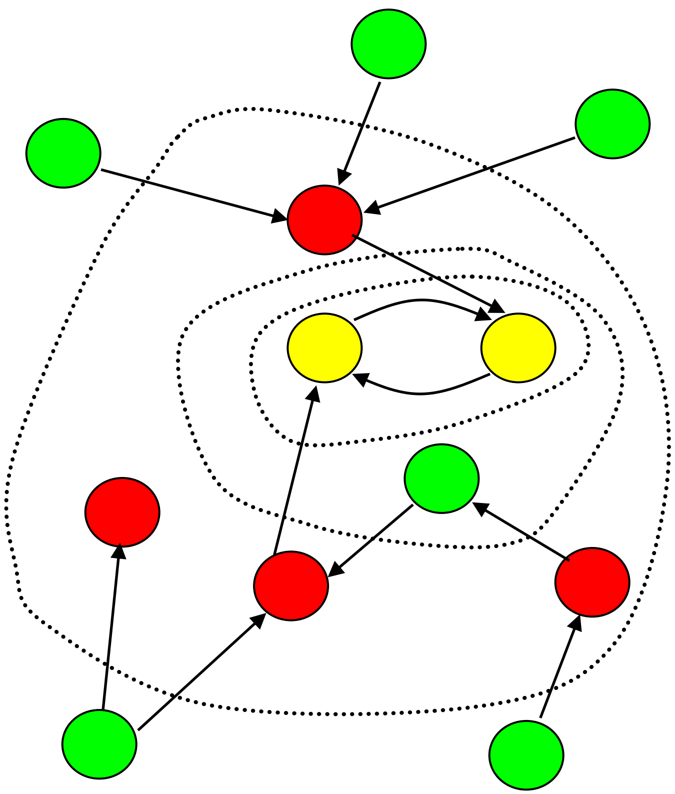
```
• for i ←1 to length[S]
• do   E'_P ← ∅
•      for each e ∈ E_P
•      do (O,I)←boundcond(Γ,S[i],e)
•         //O è il sottoinsieme dei nodi di S[i] che sono
          //attaccati da nodi ∈ e;
•         //I è il sottoinsieme dei nodi di S[i]\O che o non
          //subiscono attacchi da parte di nodi in Γ esterni a S[i] o
          //subiscono eventuali attacchi solo da parte di nodi
          //che (i) sono contenuti in Γ ma non in (S[i]∪e) e
          //(ii) sono attaccati da nodi ∈ e
•          if O = ∅
•              then if I <> ∅  then E* ←SATPref(Γ↓S[i],I∩C)
•                                //Γ↓S[i] è il sottografo di Γ che
                                 //contiene solo i nodi ∈S[i];
•                                //E* è l'insieme di tutte le PE di Γ↓S[i]
•                                 else E* ← ∅
•              else E* ←Pref(Γ↓(S[i]\O),I∩C)
•              //Γ↓(S[i]\O) è il sottografo di Γ che contiene solo i
               //nodi ∈ (S[i]\O)
•               E'_P ← E'_P ∪ (e ⊗ E*)
•              // e ⊗ E* è l'insieme di insiemi di nodi così definito:
               //{e ∪ e*| e* ∈ E*}
•       E_P ← E'_P
• return E_P
```

Algoritmo Grounded

- **Grounded**(Γ, C)   // Γ=(A,R), C⊆A

- *e* ← ∅
- I←A
- while ∃N ⊆C che contiene solo nodi che non sono attaccati da nodi in I
- do *e* ← *e* ∪ N
-     ANC ← sottoinsieme di C contenente tutti i nodi attaccati dai nodi in N
-     ANI ← sottoinsieme di I contenente tutti i nodi attaccati dai nodi in N
-     C ← C\(N ∪ANC)
-     I ← I\(N ∪ANI)
- return (e,I)

Grounded



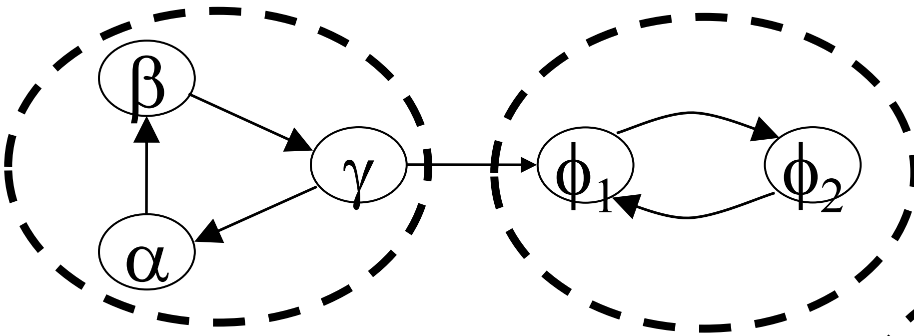
- In questo esempio si assume C=A
- Restituisce in *e* i nodi verdi e in I quelli gialli

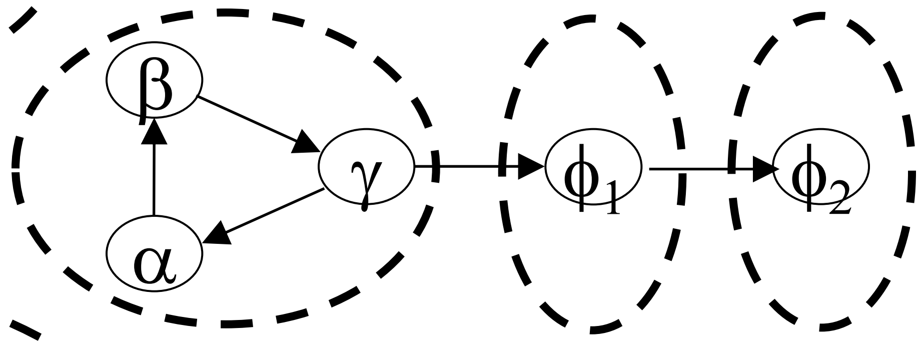
SCC (Strongly Connected Component)

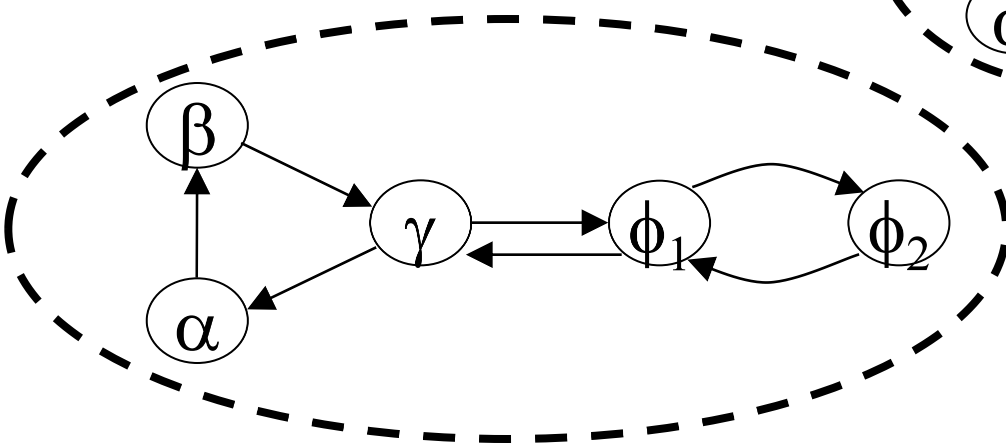
- Sottografo di un grafo orientato che gode di due proprietà:
  - partendo da ogni suo nodo, è possibile raggiungere tutti i suoi nodi (incluso quello di partenza)
  - non esiste alcun suo supergrafo che gode della proprietà precedente

SCC

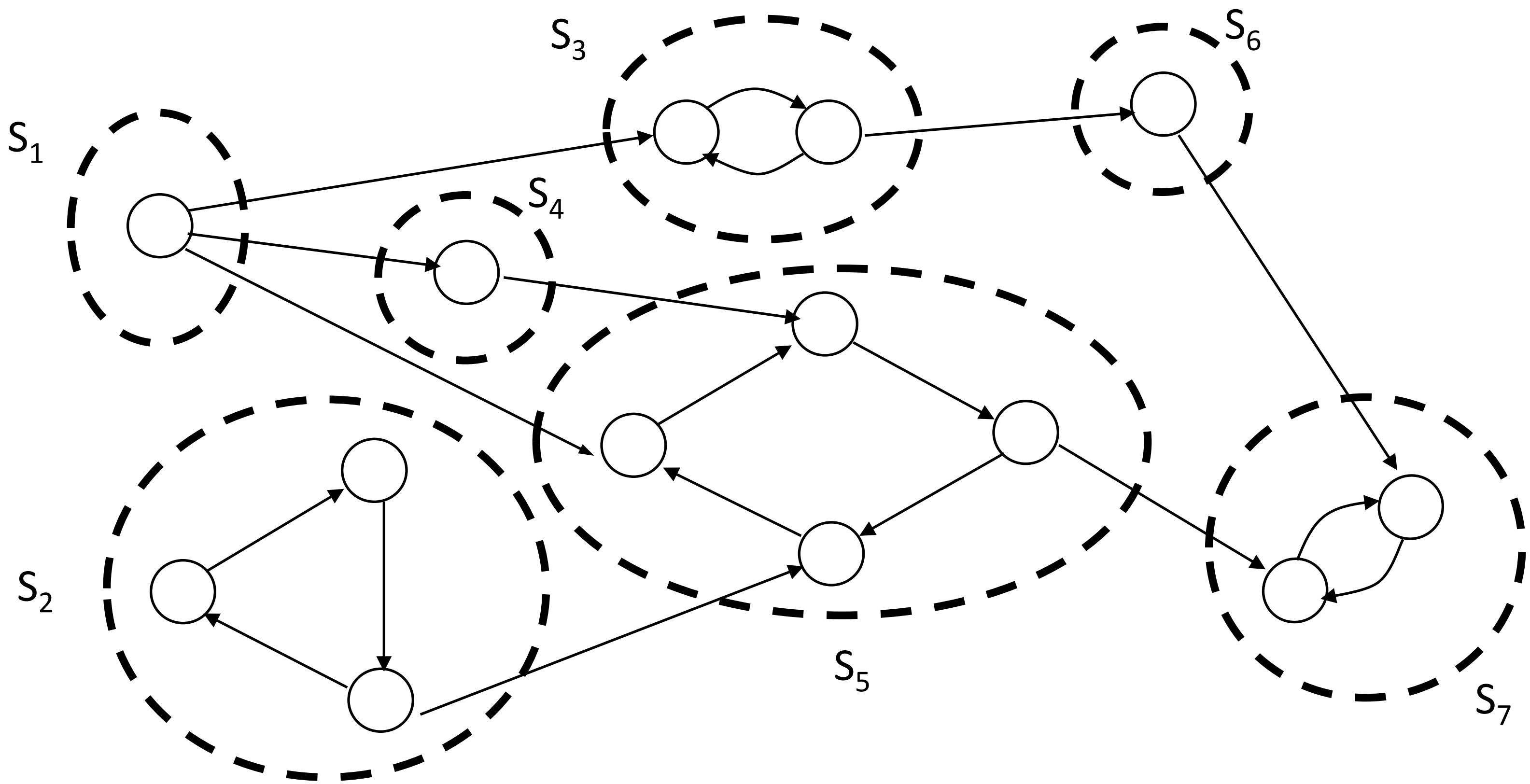
- Un grafo orientato può contenere più SCC ed è completamente coperto dalle SCC contenute





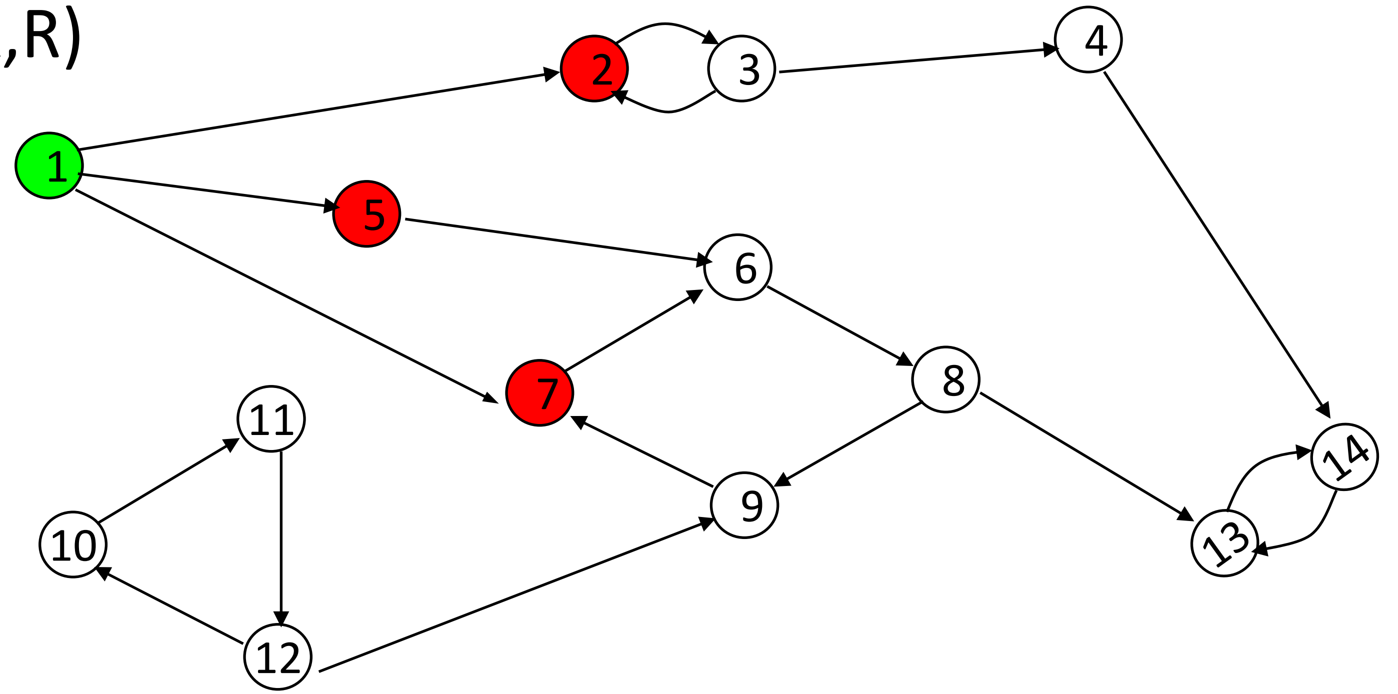


SCC

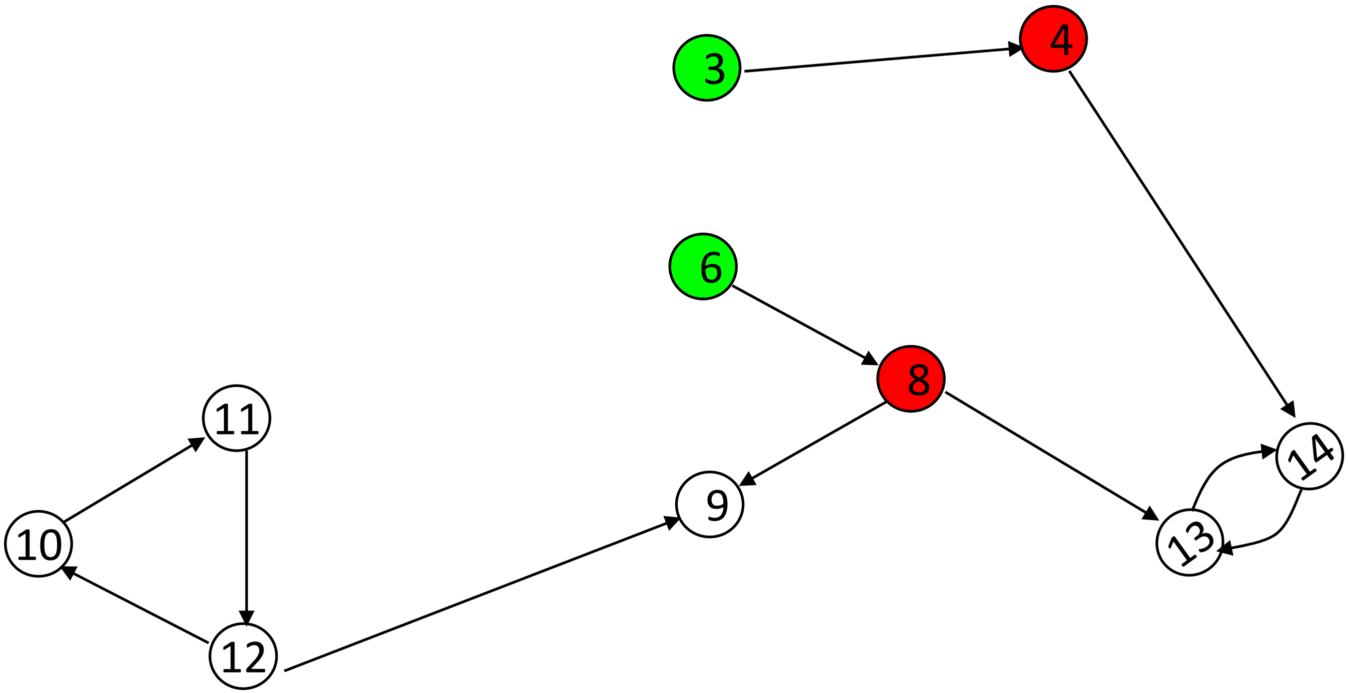


Esempio 1

Γ=(A,R)

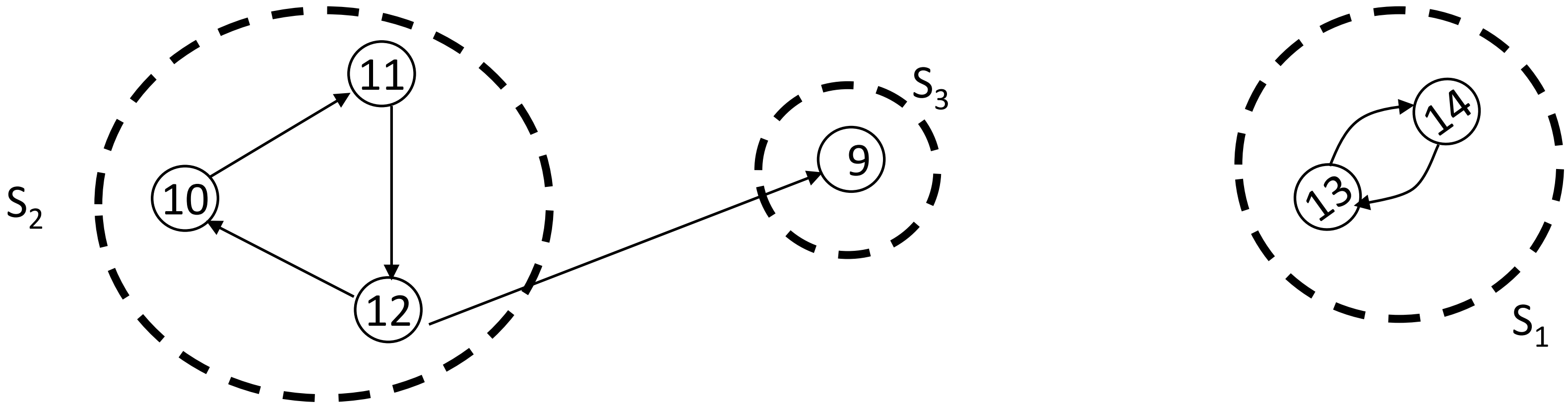


- **Pref**(Γ,C) dove C = A
- **Grounded**(Γ,{1,2,3,4,5,6,7,8,9,10,11,12,13,14})
- Prima iterazione *e*={1} , I = {3,4,6,8,9,10,11,12,13,14}

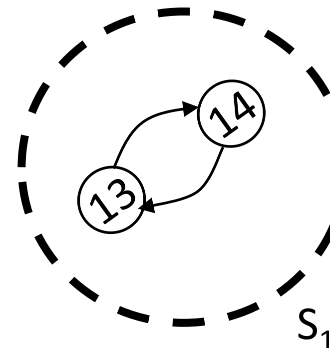


- **Grounded**, seconda (e ultima) iterazione: restituisce *e*={1,3,6}, I = {9,10,11,12,13,14}
- $E_P \leftarrow \{\{1,3,6\}\}$

$\Gamma\downarrow I$

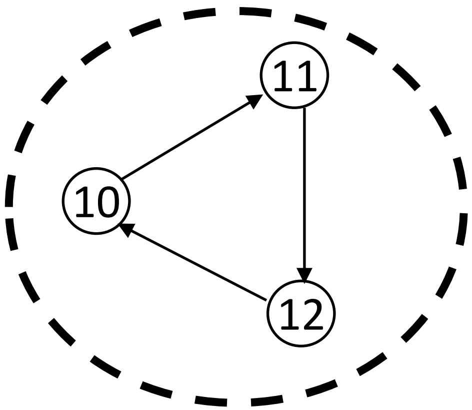


- $\Gamma \leftarrow \Gamma\downarrow I$
- Sequenza restituita da **SSCSSEQ**(Γ)
- $E'_P \leftarrow \varnothing$

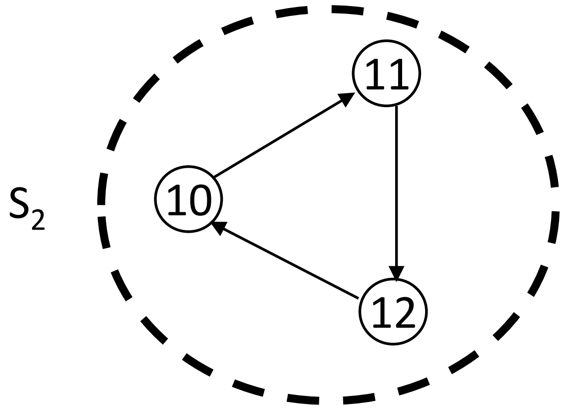


- Consideriamo i = 1 ed *e*={1,3,6}
- **boundcond** (Γ, S[1], {1,3,6}) restituisce O = $\emptyset$ e I ={13,14}
- **SATPref** ($\Gamma\downarrow$S[1],{13,14}) restituisce E* = {{13},{14}}
- $E'_P \leftarrow e \otimes E^*$= {{1,3,6,13},{1,3,6,14}}
- $E_P \leftarrow E'_P$ = {{1,3,6,13},{1,3,6,14}}
- $E'_P \leftarrow \emptyset$

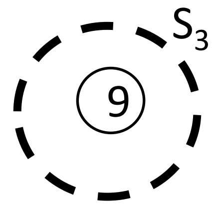
S_2



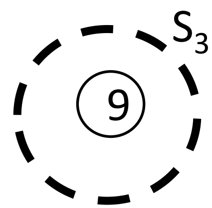
- Consideriamo i = 2 ed *e*={1,3,6,13}
- **boundcond** (Γ, S[2], {1,3,6,13}) restituisce O = $\emptyset$ e I = {10,11,12}
- **SATPref** ($\Gamma \downarrow$S[2],{10,11,12}) restituisce E* = $\emptyset$
- $E'_P \leftarrow e \otimes E^* = \{\{1,3,6,13\}\}$



- Consideriamo i= 2 ed *e*={1,3,6,14}
- **boundcond** (Γ, S[2], {1,3,6,14}) restituisce O = $\emptyset$ e I = {10,11,12}
- **SATPref** ($\Gamma \downarrow$S[2],{10,11,12}) restituisce E* = $\emptyset$
- $E'_P \leftarrow E'_P \cup (e \otimes E^*)$= {{1,3,6,13}} $\cup$ {{1,3,6,14}} = {{1,3,6,13},{1,3,6,14}}
- $E_P \leftarrow E'_P$ = {{1,3,6,13},{1,3,6,14}}
- $E'_P \leftarrow \emptyset$

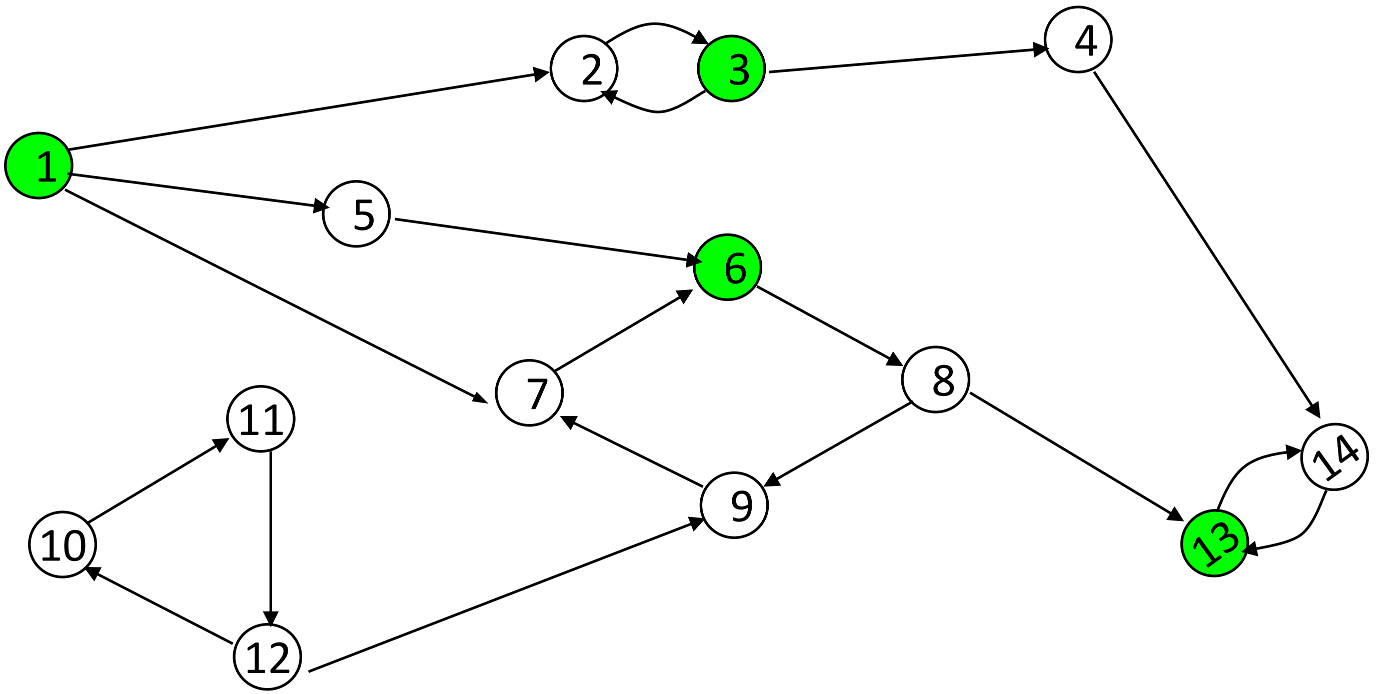


- Consideriamo i = 3 ed e={1,3,6,13}
- **boundcond** (Γ, S[3], {1,3,6,13}) restituisce O = $\emptyset$ e I = $\emptyset$
- $E^* \leftarrow \emptyset$
- $E'_P \leftarrow E'_P \cup (e \otimes E^*) = \{e\} = \{\{1,3,6,13\}\}$

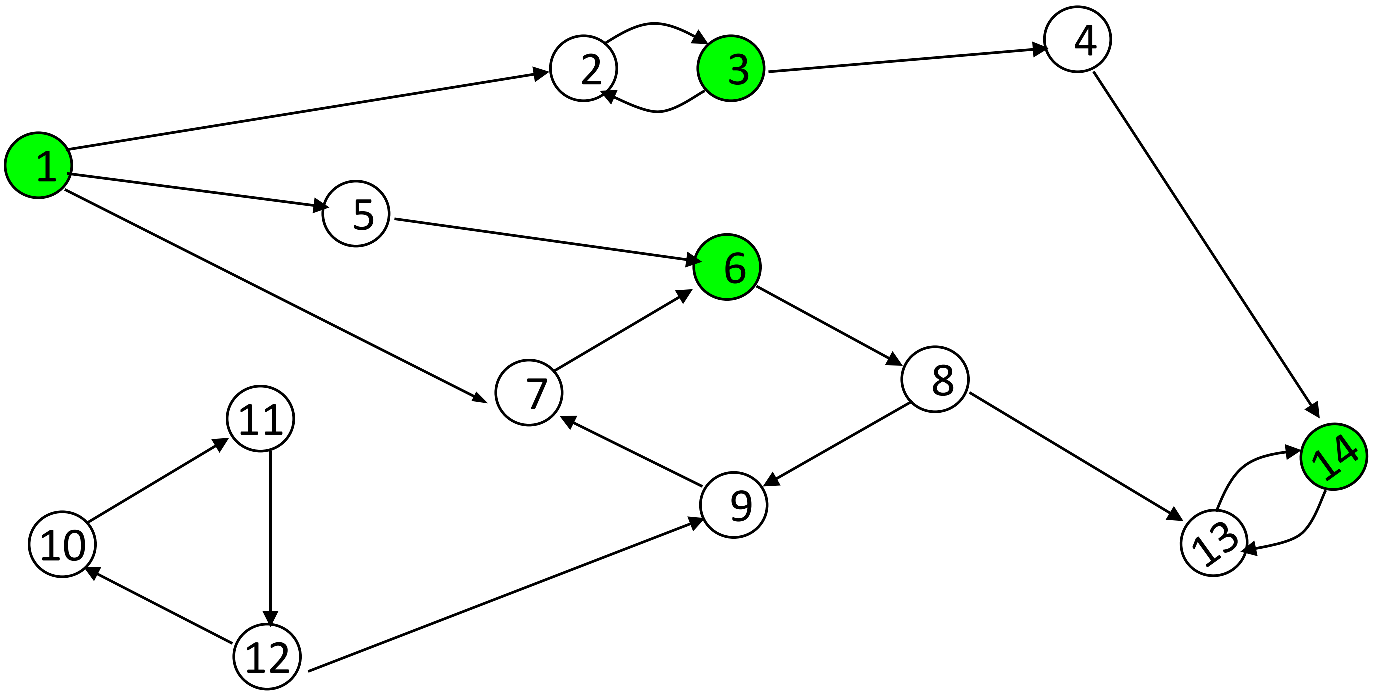


- Consideriamo i = 3 ed e={1,3,6,14}
- **boundcond** (Γ, S[3], {1,3,6,14}) restituisce O = $\varnothing$ e I = $\varnothing$
- $E^* \leftarrow \varnothing$
- $E'_P \leftarrow E'_P \cup (e \otimes E^*) = \{\{1,3,6,13\}\} \cup \{\{1,3,6,14\}\} = \{\{1,3,6,13\},\{1,3,6,14\}\}$
- $E_P \leftarrow E'_P = \{\{1,3,6,13\},\{1,3,6,14\}\}$
- return E_P

Esempio 1: prima PE in uscita

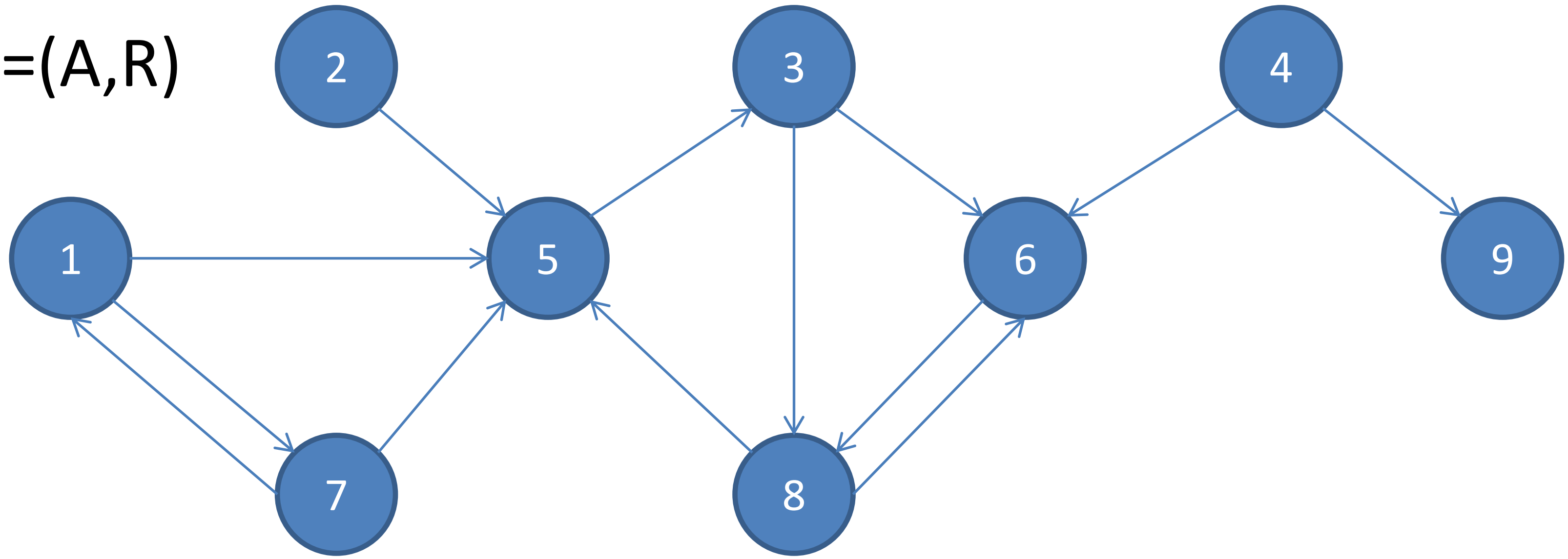


Esempio 1: seconda PE in uscita

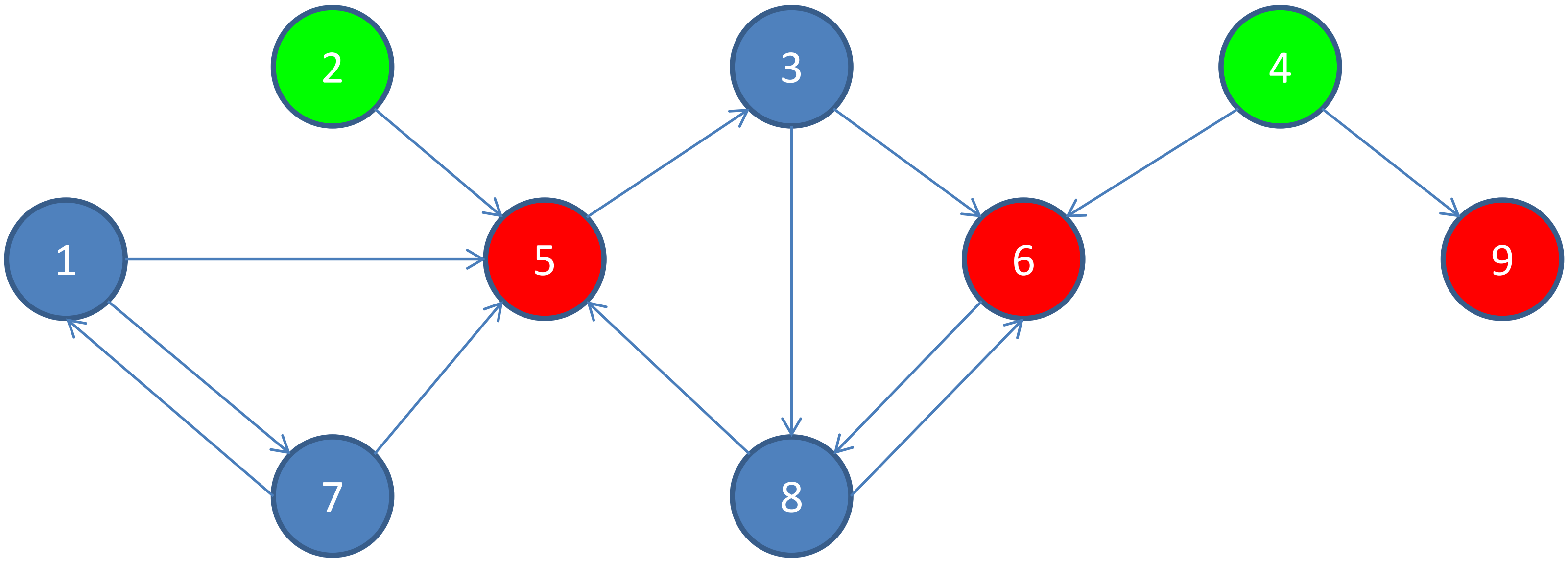


Esempio 2

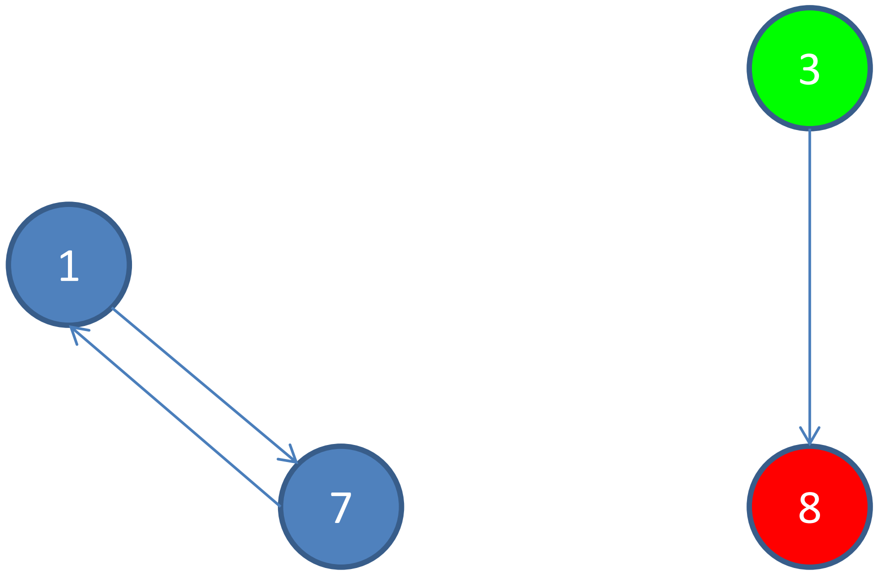
Γ=(A,R)



- **Pref**(Γ,C) dove C = A
- **Grounded**(Γ, {1,2,3,4,5,6,7,8,9})

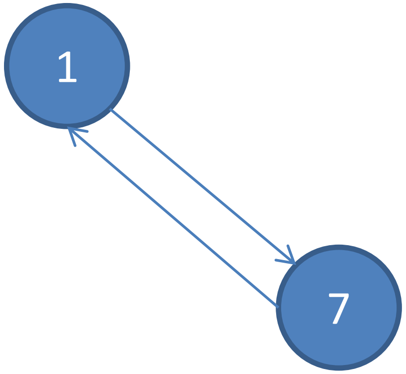


- **Grounded**: prima iterazione *e*={2,4}, C= I = {1,3,7,8}

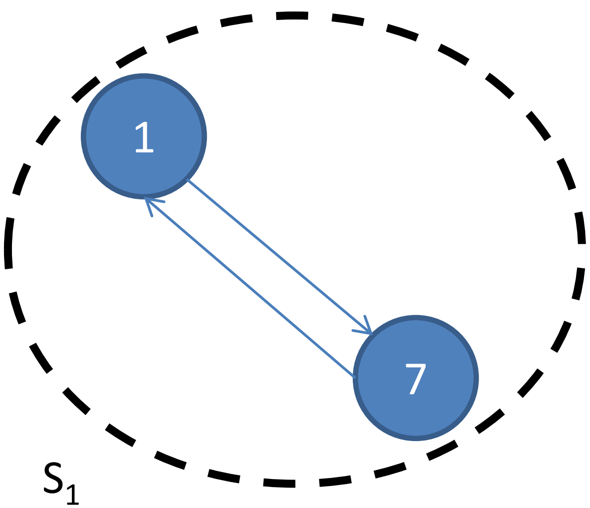


- **Grounded**: seconda (e ultima) iterazione
  e={2,3,4} ,
  C= I = {1,7}
- Restituisce *e*={2,3,4} e I = {1,7}

$\Gamma\!\downarrow\! I$



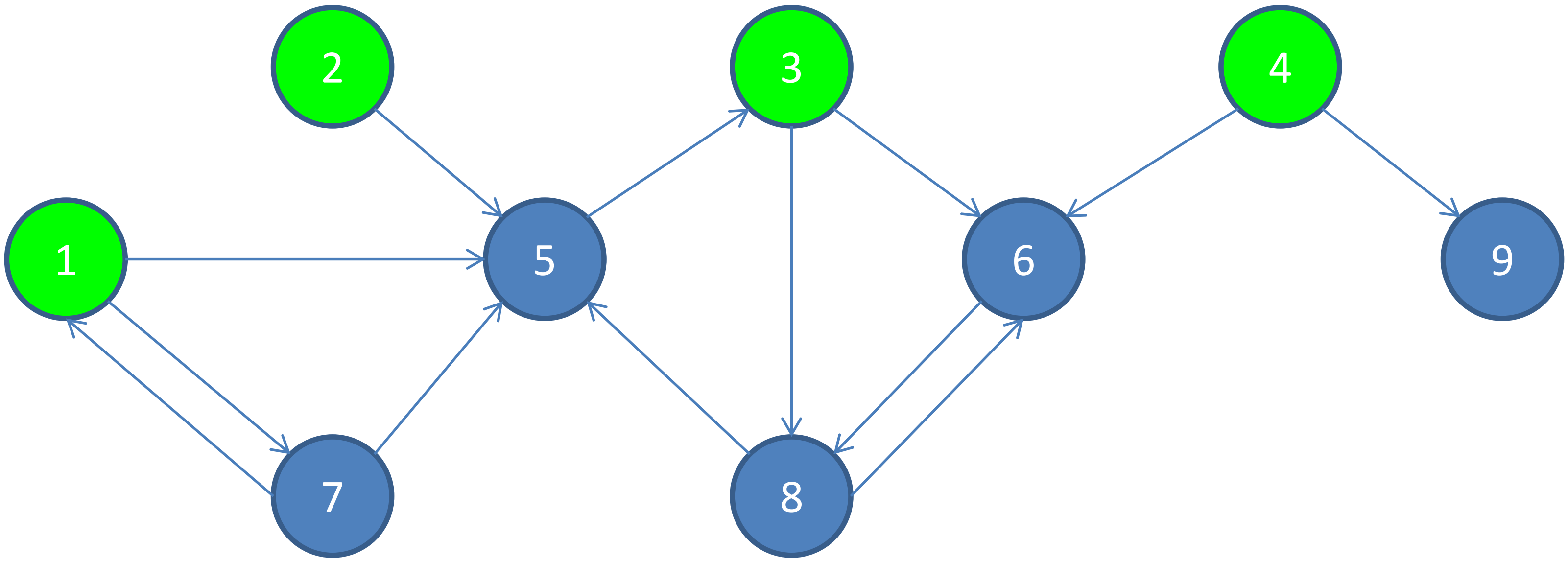
- $\Gamma \leftarrow \Gamma\!\downarrow\! I$



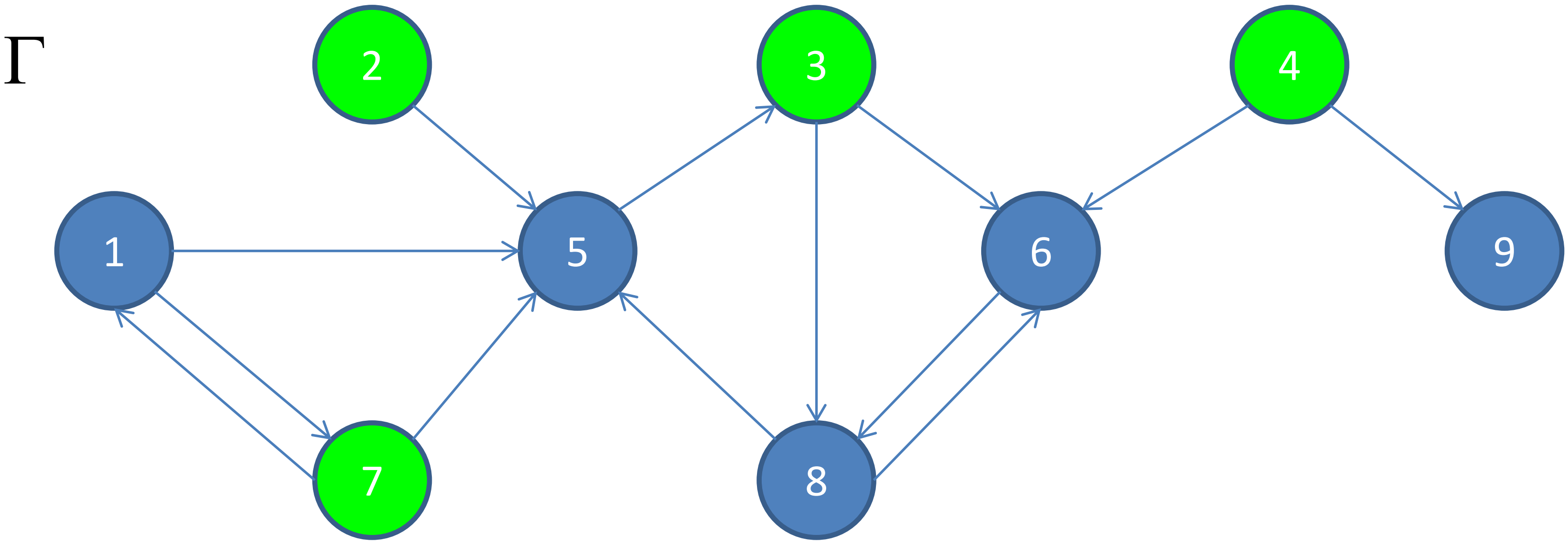
- $E_P \leftarrow \{\{2,3,4\}\}$
- Sequenza unitaria restituita da SCCSSEQ(Γ)
- $i \leftarrow 1$
- $E'_P \leftarrow \emptyset$
- $e \leftarrow \{2,3,4\}$
- **boundcond** (Γ, S[1], {2,3,4}) restituisce $O = \emptyset$ ed $I = \{1,7\}$
- **SATPref** ($\Gamma \downarrow$S[1],{1,7}) restituisce $E^* = \{\{1\},\{7\}\}$
- $E'_P \leftarrow e \otimes E^* = \{\{1,2,3,4\},\{2,3,4,7\}\}$
- $E_P \leftarrow E'_P = \{\{1,2,3,4\},\{2,3,4,7\}\}$
- return E_P

Esempio 2: prima PE in uscita

Γ

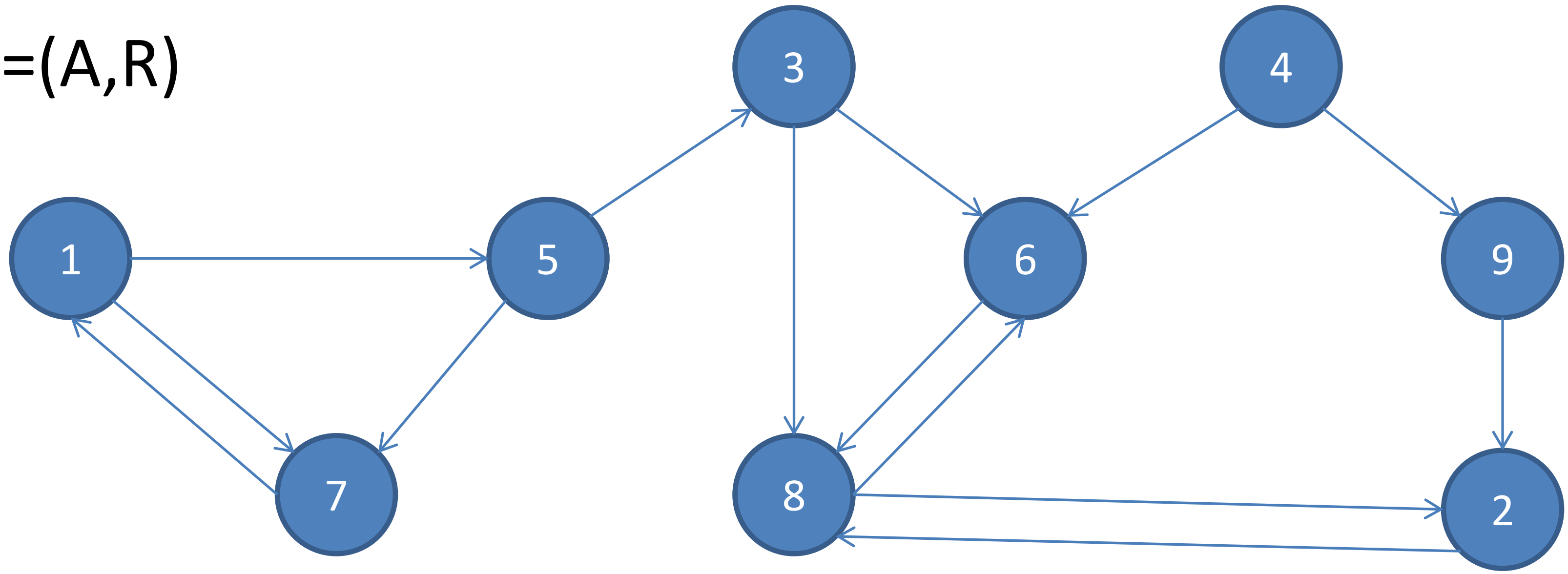


Esempio 2: seconda PE in uscita

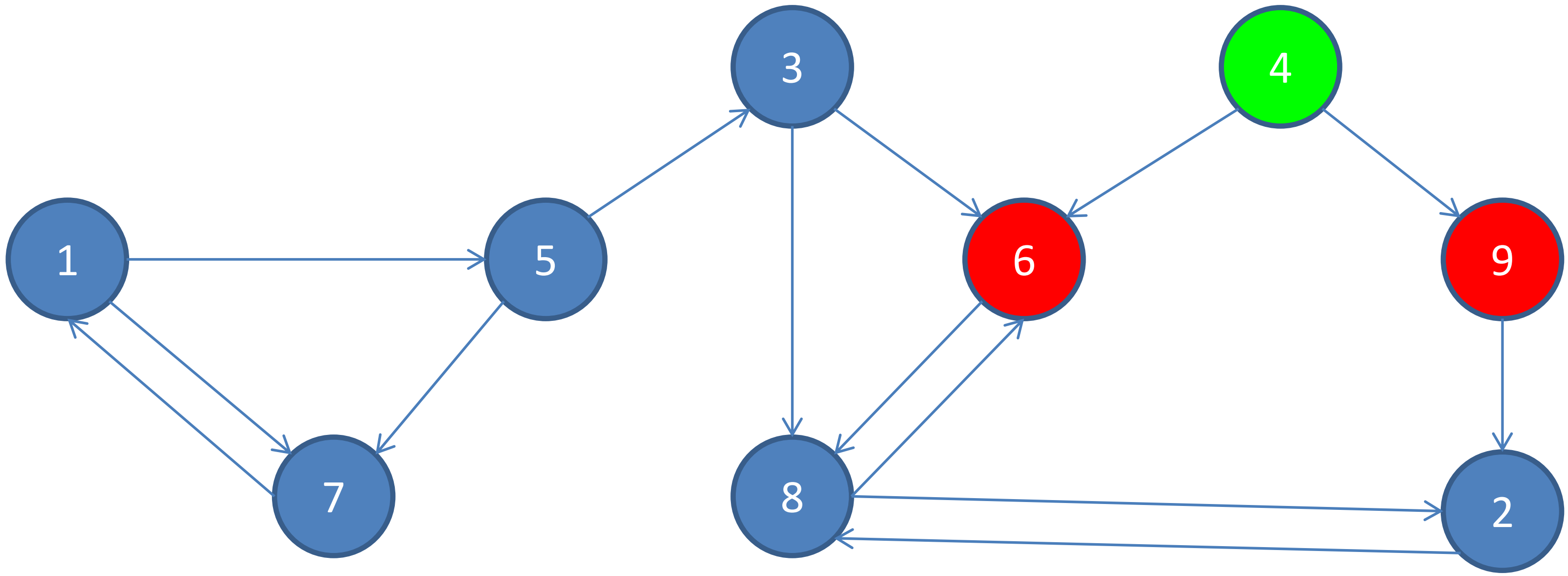


Esempio 3

Γ=(A,R)

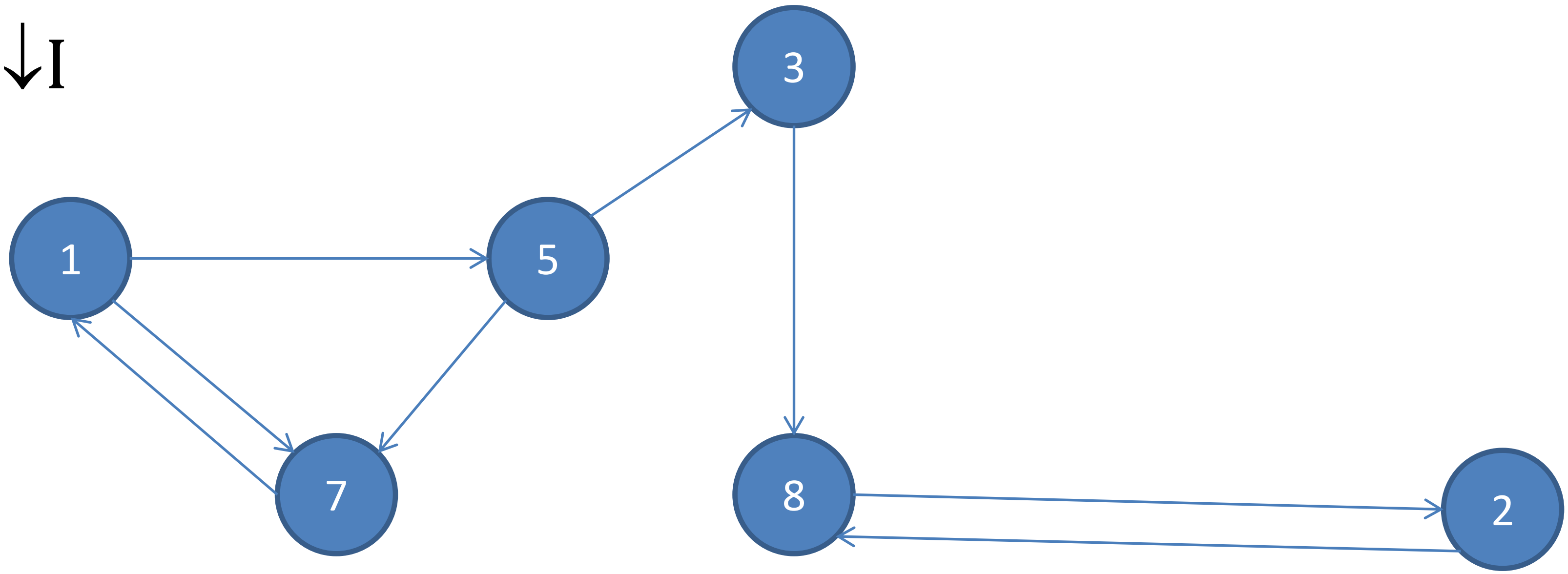


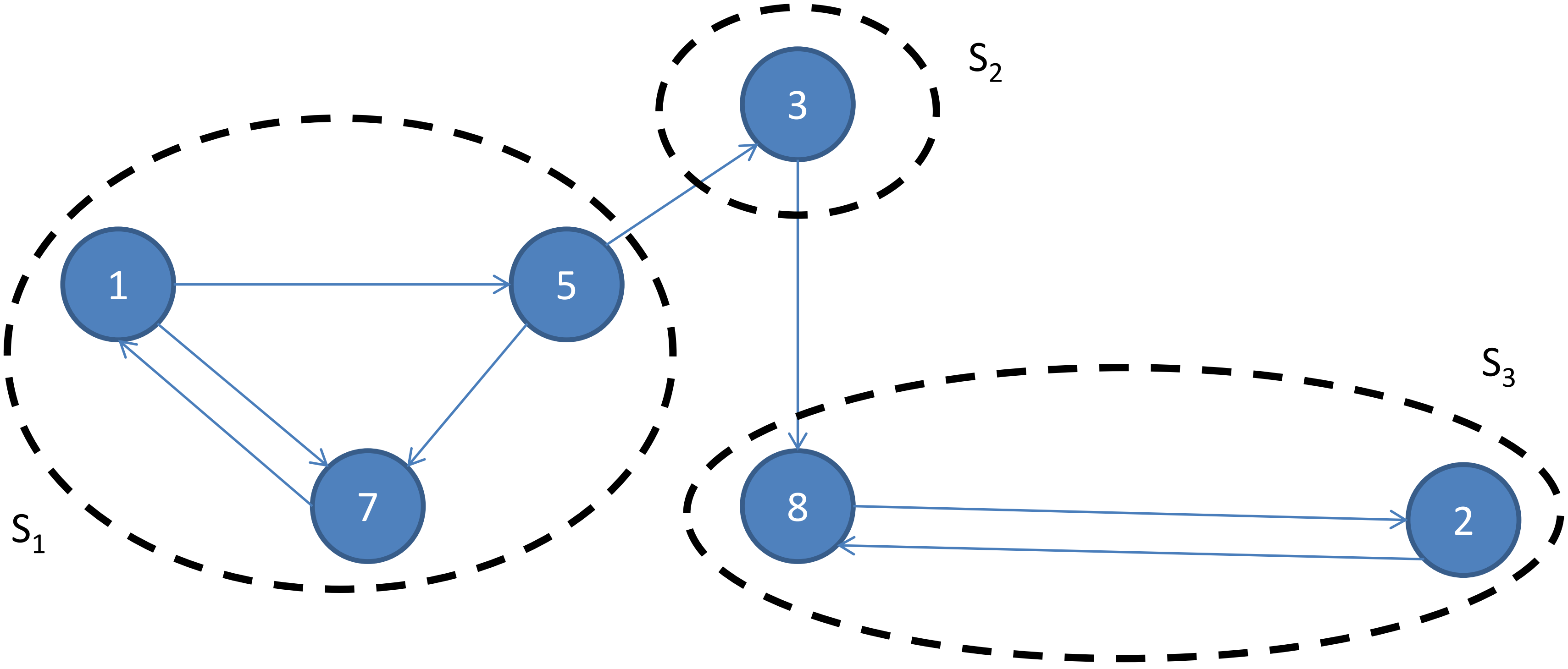
- **Pref**(Γ,C) dove C = A
- **Grounded**(Γ, {1,2,3,4,5,6,7,8,9})



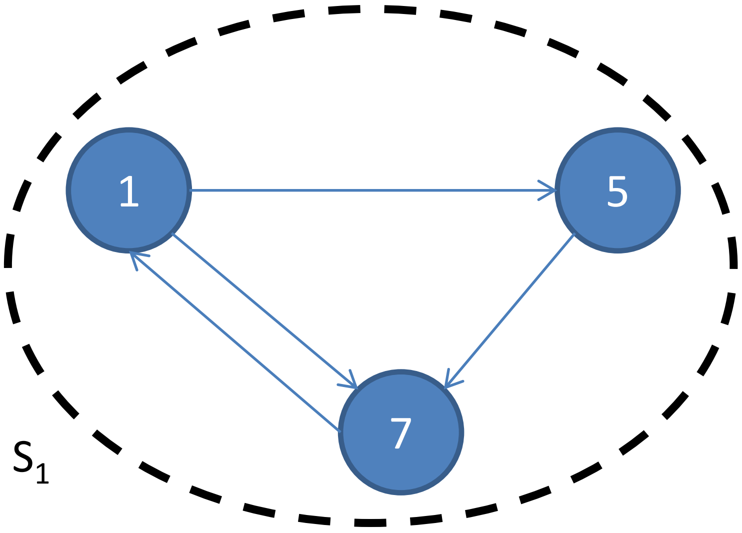
- **Grounded**: prima (e ultima) iterazione *e*={4} , C= I = {1,2,3,5,7,8}
- Restituisce *e*={4} , C= I = {1,2,3,5,7,8}
- $E_P \leftarrow$ {{4}}

$\Gamma{\downarrow}\mathrm{I}$

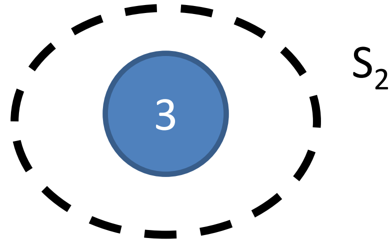




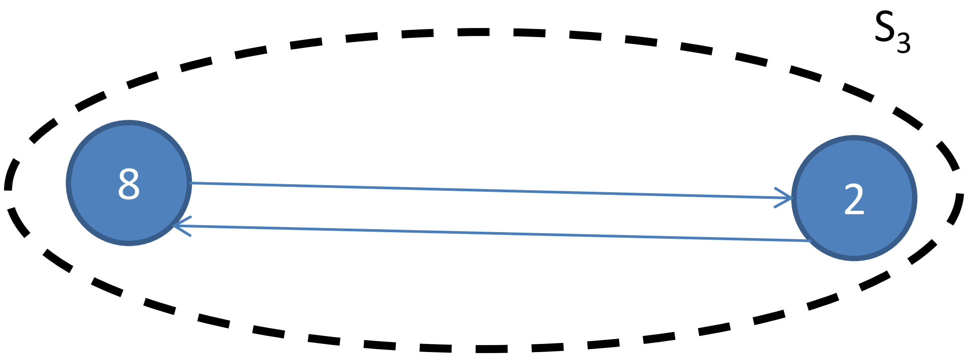
- $\Gamma \leftarrow \Gamma\!\downarrow\! I$
- Sequenza restituita da SCCSSEQ(Γ)



- $i \leftarrow 1$
- $E'_P \leftarrow \varnothing$
- $e \leftarrow \{4\}$
- **boundcond** (Γ, S[1], {4}) restituisce O = $\varnothing$ e I = {1,5,7}
- **SATPref** ($\Gamma \downarrow$S[1],{1,5,7}) restituisce E* = {{1}}
- $E'_P \leftarrow e \otimes E^* = \{\{1, 4\}\}$
- $E_P \leftarrow E'_P = \{\{1,4\}\}$

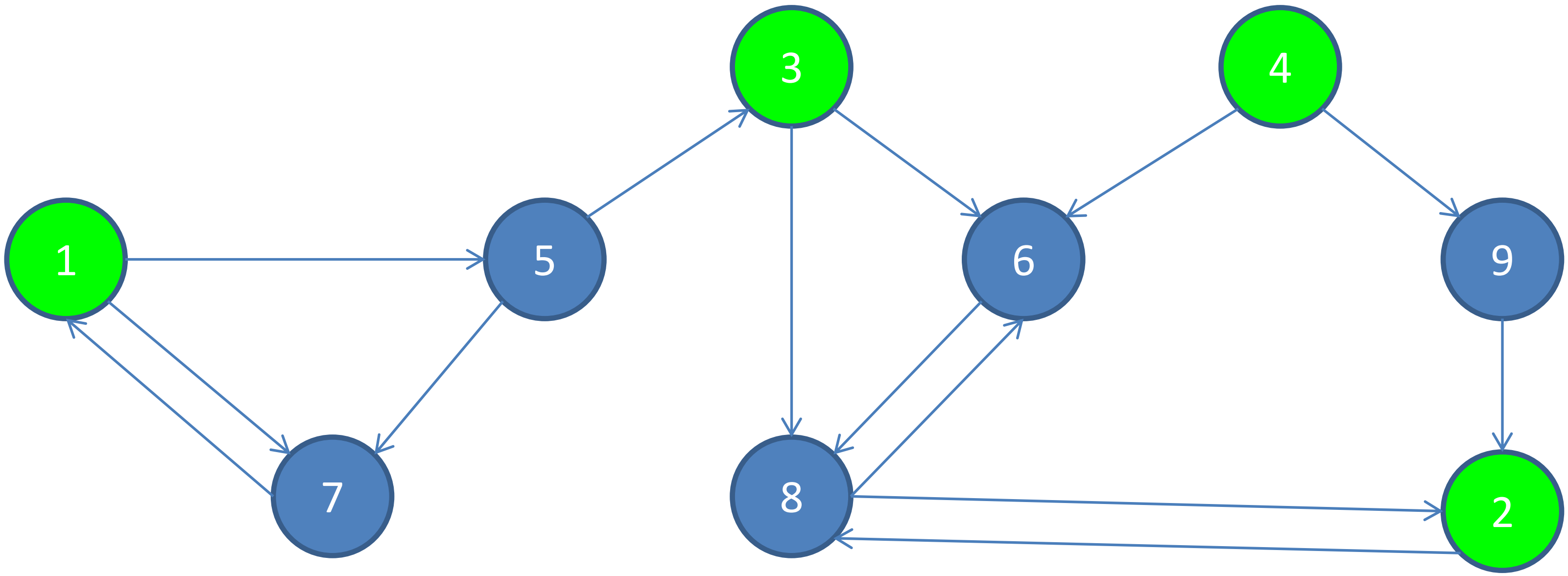


- $i \leftarrow 2$
- $E'_P \leftarrow \varnothing$
- $e \leftarrow \{1,4\}$
- **boundcond** $(\Gamma$, S[2], {1,4}) restituisce $O = \varnothing$ e $\mathrm{I} = \{3\}$
- **SATPref** $(\Gamma \downarrow S[2], \{3\})$ restituisce $E^* = \{\{3\}\}$
- $E'_P \leftarrow e \otimes E^* = \{\{1, 3,4\}\}$
- $E_P \leftarrow E'_P = \{\{1,3,4\}\}$



- i $\leftarrow$ 3
- $E'_P \leftarrow \varnothing$
- $e \leftarrow \{1,3,4\}$
- **boundcond** $(\Gamma$, S[3], {1,3,4}) restituisce O = {8} e I = {2}
- **Pref** $(\Gamma\downarrow(S[3]\backslash O),\{2\})$
- **Grounded**$(\Gamma\downarrow(S[3]\backslash O),\{2\})$ restituisce e ={2} e $I = \varnothing$
- $E_P \leftarrow \{\{2\}\}$
- return $E_P = \{\{2\}\}$
- $E^* = \{\{2\}\}$
- $E'_P \leftarrow e \otimes E^* = \{\{1,2,3,4\}\}$
- $E_P \leftarrow E'_P = \{\{1,2,3,4\}\}$
- return $E_P = \{\{1,2,3,4\}\}$

Esempio 3: unica PE in uscita



Richieste

- Ogni gruppo di due studenti deve implementare l'algoritmo ricorsivo **Pref**
- Ciò richiede la codifica di tre algoritmi che esso invoca, cioè
  - **Grounded** (di cui è fornito lo pseudocodice),
  - **SCCSSEQ** (il gruppo deve scegliere in letteratura un algoritmo finalizzato all'individuazione degli SCC di un grafo e un altro algoritmo per l'ordinamento topologico degli SCC stessi),
  - **boundcond** (di cui sono fornite le specifiche in linguaggio naturale sotto forma di commenti nello pseudocodice di **Pref**),

  mentre dell'algoritmo invocato **SATPref** è fornita l'implementazione
- Effettuare scelte delle strutture dati volte a ridurre i tempi di esecuzione di **Pref**, documentando le stesse
- Verificare il codice, documentando i risultati
- Ogni intervento volto a migliorare l'efficienza dell'algoritmo è benvenuto

Qualche miglioramento

- Tutte le chiamate di **boundcond** che inducono al calcolo dei medesimi O ed I sono necessariamente seguite dal calcolo del medesimo E* $\rightarrow$ evitare di ricalcolare il medesimo E* (sfruttare un trade-off spazio-tempo)
- Talvolta si sa a priori che una chiamata di **boundcond** induce al calcolo dei medesimi O ed I già determinati da una chiamata precedente $\rightarrow$ evitare di ricalcolare O ed I (vedi anche lucido successivo)

Su **boundcond**

- Si considerino le due chiamate
  (O_a, I_a) $\leftarrow$**boundcond**(Γ,S[i],e_a) ed
  (O_b, I_b) $\leftarrow$**boundcond**(Γ,S[i],e_b).

  Per definizione sia α[i] l'unione (dei nodi) degli SCC padri di S[i], dove S[j] si dice *padre* di S[i] se attacca S[i] . (Si noti che tutti i padri di S[i] precedono S[i] secondo l'ordinamento topologico – ma non tutti gli SCC che precedono S[i] secondo tale ordinamento sono sui padri.)

  Se $e_a \cap \alpha[i] = e_b \cap \alpha[i]$, allora $O_a = O_b$.

- Una conseguenza del punto precedente è che, se S[i-1] non è padre di S[i], tutte le invocazioni
  (O, I) $\leftarrow$**boundcond**(Γ,S[i],e)
  effettuate entro il ciclo **for** annidato restituiscono lo stesso insieme O per ogni $e \in E_P$.

Su **boundcond** (cont.)

- Altra conseguenza è che, nella invocazione principale di **Pref** così come per ogni invocazione ricorsiva di **Pref** in cui l'insieme dei nodi del grafo che costituisce il primo parametro coincida con l'insieme che costituisce il secondo parametro, per ogni S[i] privo di padri (cioè tale che α[i] = $\emptyset$), a ogni iterazione del ciclo **for** più annidato **boundcond** restituisce necessariamente O = $\emptyset$ (perché nessun nodo di tali SCC è attaccato da *e*) ed I = S[i] (dal momento che nessun nodo di S[i] è attaccato da nodi esterni a S[i]) $\rightarrow$ evitare la chiamata di **boundcond**, sfruttando direttamente O = $\emptyset$ e I = S[i]

Qualche altro miglioramento

- Evitare di invocare **SATPref** se il primo parametro coincide col secondo e si tratta di una SCC contenente uno o due nodi perché si sa che l'insieme restituito nel primo caso contiene solo un singoletto che contiene l'unico nodo e nel secondo contiene due singoletti, uno per ciascun nodo
- Evitare di invocare ricorsivamente **Pref** o di invocare **Grounded** se il primo parametro coincide col secondo e si tratta di un grafo contenente un solo nodo, perché si sa che l'insieme restituito contiene solo un singoletto che contiene l'unico nodo

Requisiti non funzionali

- Linguaggio di programmazione: C++
  (il codice **SATPref** fa riferimento a librerie stardard C++)
- IDE consigliato: Eclipse

- L'interfacciamento con **SATPref** richiede di utilizzare le classi dei parametri passati (di input e/o output): si dovrà riusare il codice di tali classi, che sarà fornito unitamente a metodi di utilità