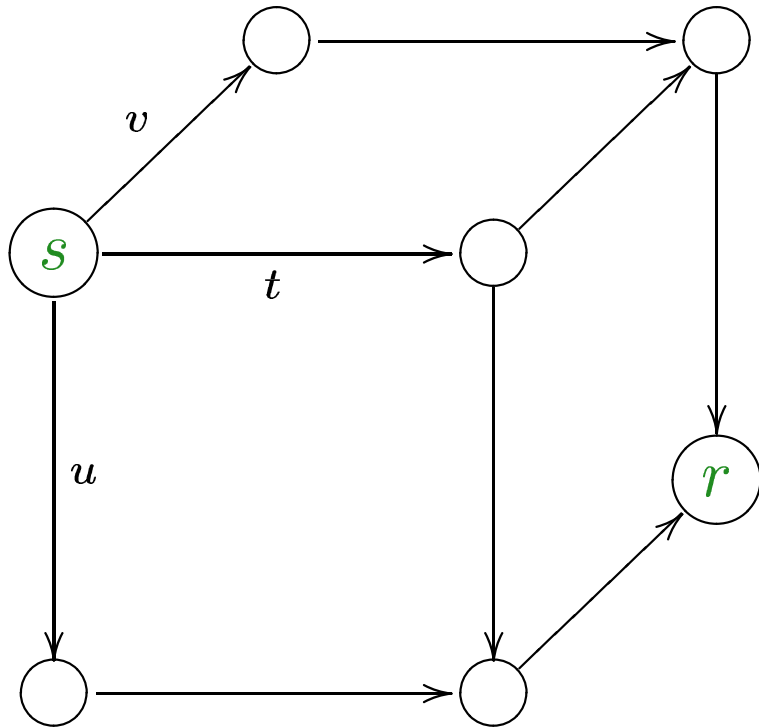


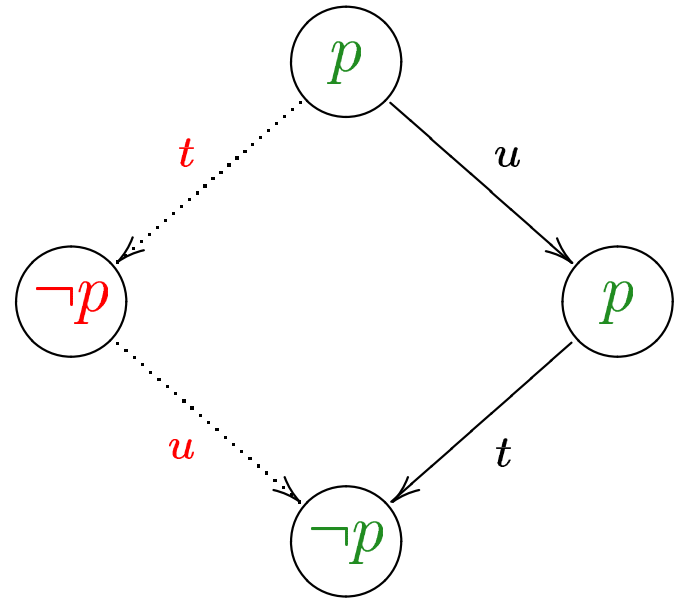
# Praktyczne metody weryfikacji

## Wykład 5: Redukcje cz.-p.

```
proc dfs(s)
  if error(s) then report error fi
  add {s,0} to Statespace
  add s to Stack
  for each (selected) successor t of s do
    if {t,0} not in Statespace then dfs(t) fi
  od
  if accepting(s) then ndfs(s) fi
  delete s from Stack
end
proc ndfs(s) /* the nested search */
  add {s,1} to Statespace
  for each (selected) successor t of s do
    if {t,1} not in Statespace then ndfs(t) fi
    else if t in Stack then report cycle fi
  od
end
```



$F \neg p$



**Def.:**  $M = \langle S, S_{\text{pocz}}, T, L \rangle$   $T$  – operacje (tranzycje)

dla  $\alpha \in T$ :  $\text{en}_\alpha \subseteq S$ ,  $\alpha : \text{en}_\alpha \rightarrow S$

**ścieżka:**  $\Pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} \dots$   $s_0 = s_{\text{pocz}}$

$\alpha_i(s_i) = s_{i+1}$

$\text{en}_s := \{\alpha \mid s \in \text{en}_\alpha\}$   $(\alpha \in \text{en}_s \iff s \in \text{en}_\alpha)$

**Pomysł:**  $\text{ample}_s \subseteq \text{en}_s$  zamiast  $\text{en}_s$  w podwójnym DFS'ie ?

**Pomysł:**  $\text{ample}_s \subseteq \text{en}_s$  zamiast  $\text{en}_s$  w podwójnym DFS'ie ?

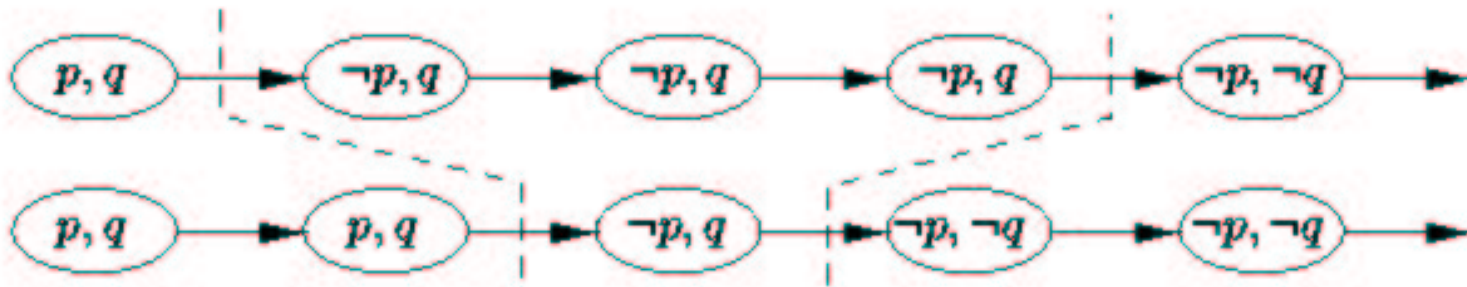
To ma sens, gdy:

- weryfikacja daje ten sam wynik (poprawność)
- znacząco maleje liczba stanów
- narzut czasowy jest rozsądny (opłacalność)

**Def.:**  $\Pi = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$  i  $\Pi' = s'_0 \rightarrow s'_1 \rightarrow s'_2 \rightarrow \dots$  są równoważne (ang. stuttering equivalence),  $\Pi \equiv \Pi'$ , jeśli ciągi

$$L(s_0), L(s_1), L(s_2), \dots \quad L(s'_0), L(s'_1), L(s'_2), \dots$$

stają się identyczne po pogrupowaniu:



**Def.:**  $M \equiv M'$  wtw. gdy

- $\forall \Pi w M \exists \Pi' w M' \Pi \equiv \Pi'$
- $\forall \Pi' w M' \exists \Pi w M \Pi \equiv \Pi'$

LTL<sub>X</sub> = LTL bez X

**Tw.:** Gdy  $\phi \in \text{LTL}_X$  i  $\Pi \equiv \Pi'$ , to  $\Pi \models \phi \iff \Pi' \models \phi$

**Tw.:** Gdy  $\phi \in \text{LTL}_X$  i  $M \equiv M'$ , to  $M \models \phi \iff M' \models \phi$

**Tw.:** LTL<sub>X</sub> = FO<sub>≡</sub>

**Poprawno:**

$$M \xrightarrow{\text{redukcja}} M'$$

$$M \equiv M'$$



**Def.:** Relacja niezależności  $I \subseteq T \times T$ :

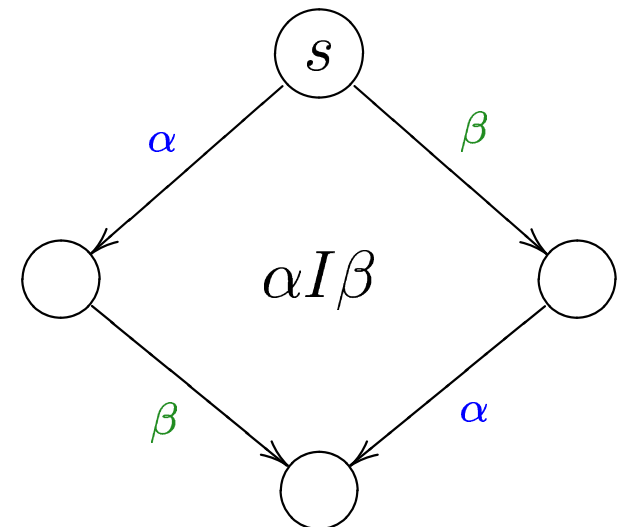
– relacja przeciwzwrotna i symetryczna

– jeśli  $\alpha I \beta$ ,  $\alpha \in \text{en}_s$ ,  $\beta \in \text{en}_s$ , to

$(s \in \text{en}_\alpha \cap \text{en}_\beta)$

–  $\beta(s) \in \text{en}_\alpha$ ,  $\alpha(s) \in \text{en}_\beta$

–  $\beta(\alpha(s)) = \alpha(\beta(s))$



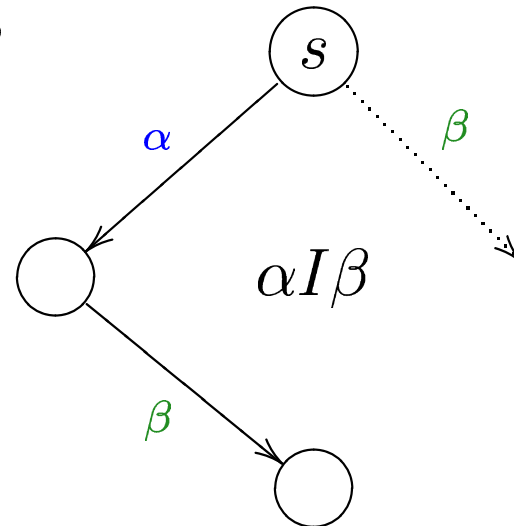
$D = T \times T \setminus I$  (relacja zależności)

**Przykład:** Niezależne **mogą być:**

- 2 instrukcje różnych procesów operujące na zmiennych lokalnych
- 2 instrukcje różnych procesów inkrementujące tę samą zmienną globalną
- 2 instrukcje różnych procesów piszące lub czytające z różnych buforów
- 2 instrukcje **tego samego procesu** ?

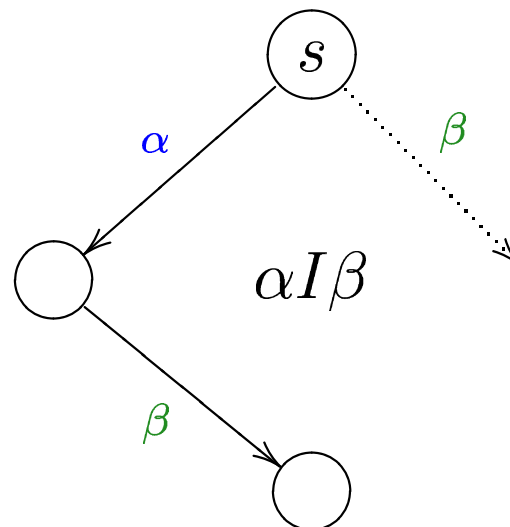
**Pytanie:** Niech  $\alpha I \beta$ . Czy możliwe jest

$$s \in \text{en}_\alpha \setminus \text{en}_\beta \quad \alpha(s) \in \text{en}_\beta ?$$



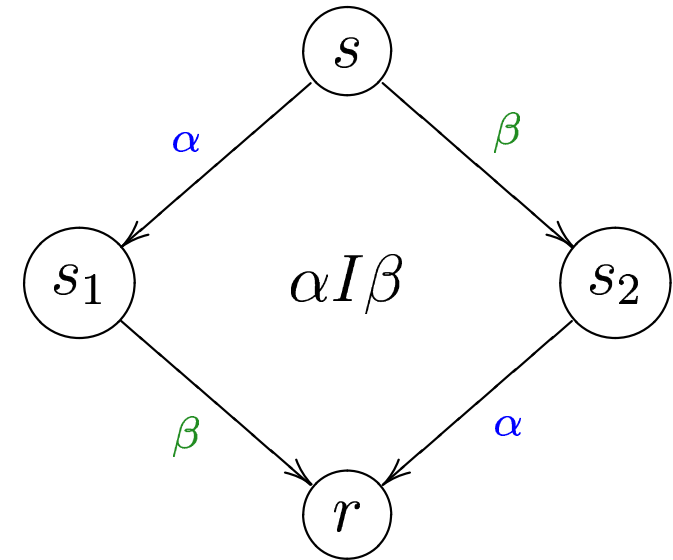
**Pytanie:** Niech  $\alpha I \beta$ . Czy możliwe jest

$$s \in \text{en}_\alpha \setminus \text{en}_\beta \quad \alpha(s) \in \text{en}_\beta ?$$



**Tak!** Np. czytanie i pisanie do tego samego bufora asynchronicznego.

Kiedy można ignorować  $s_1$  lub  $s_2$  ?



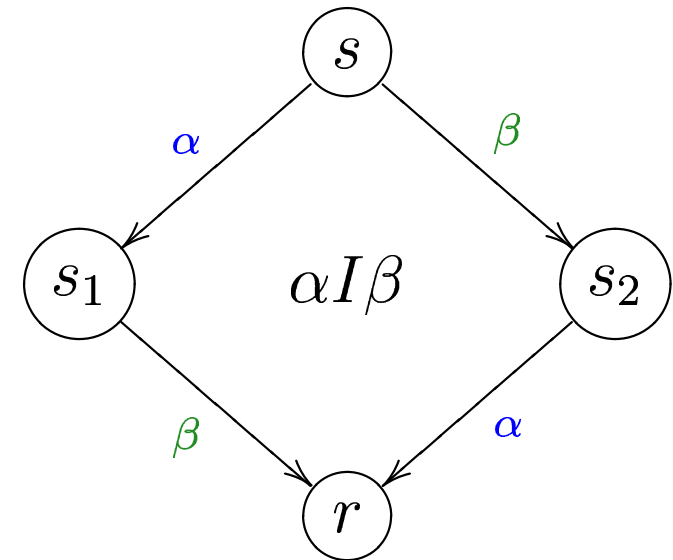
**Problem 1:** Własność  $\phi$  może zależeć od  $s_1$  i od  $s_2$ .

**Problem 2:** stan  $s_1$  ( $s_2$ ) może mieć następniki nieosiągalne inaczej.

**Def.:**  $\alpha$  jest **niewidoczna** gdy  $L(s) = L(\alpha(s)), \forall s \in \text{en}_q$ .

**Przykład:** Jeśli  $\alpha$  niewidoczna, to

$$ss_1r \equiv ss_2r$$



## Warunki (C0) – (C3)

(C0)  $\text{ample}_s = \emptyset \iff \text{en}_s = \emptyset$

(C1) ...

(C2) jeśli  $\text{ample}_s \neq \text{en}_s$ , to każda  $\alpha \in \text{ample}_s$  jest  
niewidoczna

(C3) ...

**(C1)** tranzycja zależna od pewnej tranzycji z  $\text{ample}_s$   
nie może być wykonana zanim nie wykona się E  
tranzycja z  $\text{ample}_s$

**(C1)** dla każdej ścieżki  $\Pi$  rozpoczynającej się z  $s$ :

jeśli  $\alpha \in \text{ample}_s$ ,  $\beta \notin \text{ample}_s$ ,  $\alpha D \beta$

to  $\beta$  nie może być wykonane w  $\Pi$

zanim nie wykona się pewna tranzycja z  $\text{ample}_s$



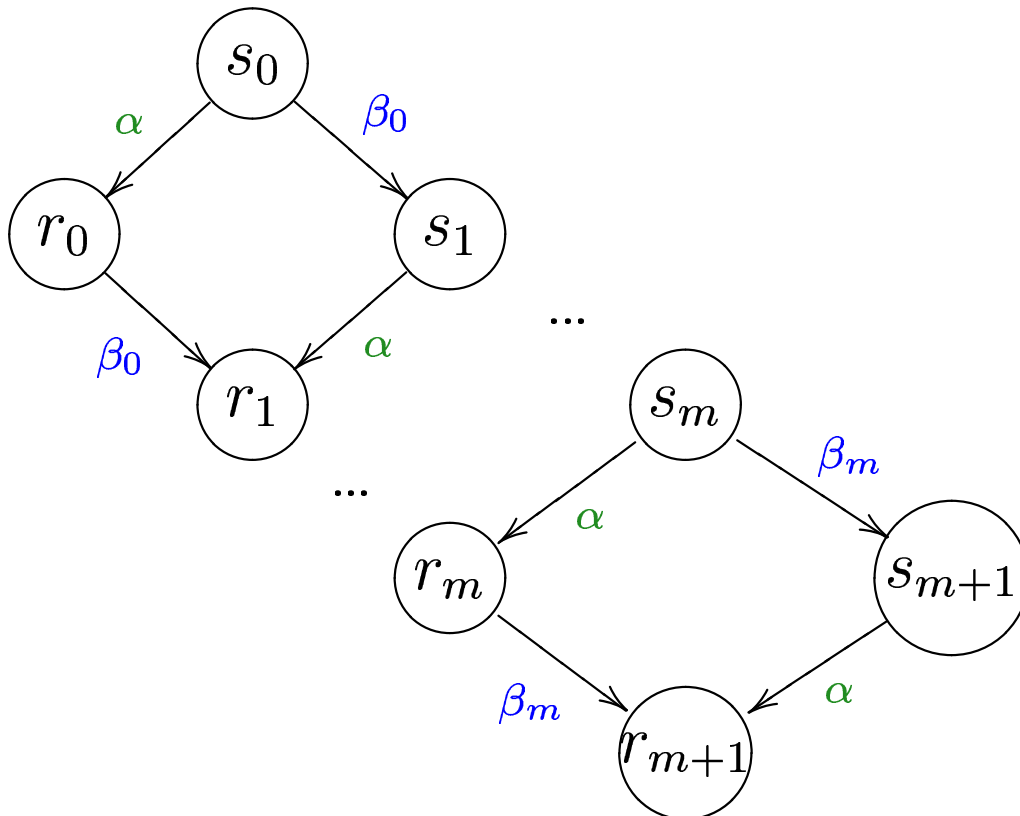
**Lemat:** Jeśli **(C1)**, to  $(\text{en}_s \setminus \text{ample}_s) \perp \text{ample}_s$ .

**Dowód:** Niech  $\beta \in \text{en}_s \setminus \text{ample}_s$ ,  $\alpha \in \text{ample}_s$ ,  $\alpha D\beta$ .

$s \xrightarrow{\beta} \beta(s) \rightarrow \dots$       **sprzeczność z (C1)**.

I.  $s = s_0 \xrightarrow{\beta_0} s_1 \xrightarrow{\beta_1} \dots s_m \xrightarrow{\beta_m} s_{m+1} \xrightarrow{\alpha} \dots$   $\alpha \in \text{ample}_s$

II.  $s = s_0 \xrightarrow{\beta_0} s_1 \xrightarrow{\beta_1} \dots$   $\forall i \beta_i \perp \alpha$

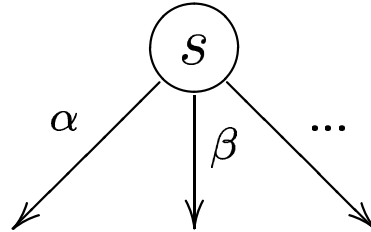


I.  $\beta_0 \dots \beta_m \alpha \equiv \alpha \beta_0 \dots \beta_m$

II.  $\beta_0 \dots \equiv \gamma \beta_0 \dots$

$\gamma \in \text{ample}_s$  dowolne

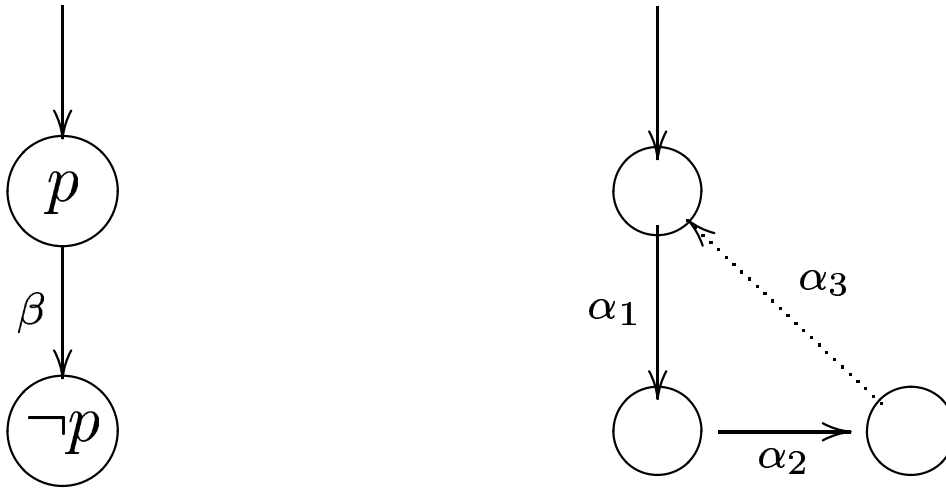
**Def. (sprawiedliwość):** dla każdego osiągalnego stanu  $s$ , jeśli  $\alpha \in \text{en}_s$  to kiedyś musi zostać wykonana  $\beta$  t. że  $\alpha D \beta$ .



Przypadek II. niemożliwy, gdy założymy sprawiedliwość

Czy (C0) – (C2) są wystarczające?

Nie!

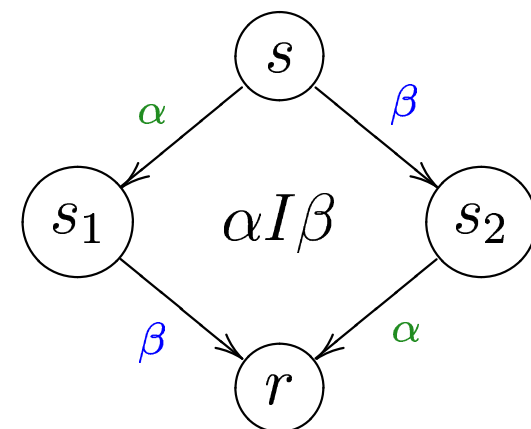


(C3) zabraniamy cykli t. że  $\exists \beta$  t. że

$\forall s$  w cyklu,  $\beta \in \text{en}_s \setminus \text{ample}_s$

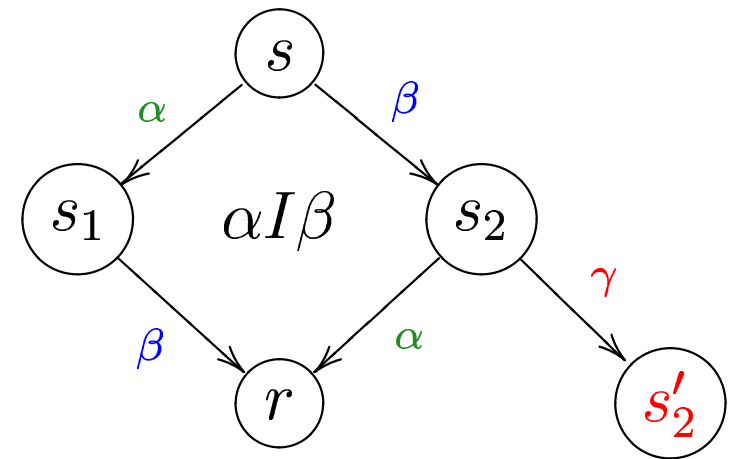
**Problem 1:** Własność  $\phi$  może zależeć od  $s_1$  i od  $s_2$ .

Rozwiązany dzięki **(C2)** .



**(C2)** jeśli  $\text{ample}_s \neq \text{en}_s$ , to każda  $\alpha \in \text{ample}_s$  jest niewidoczna

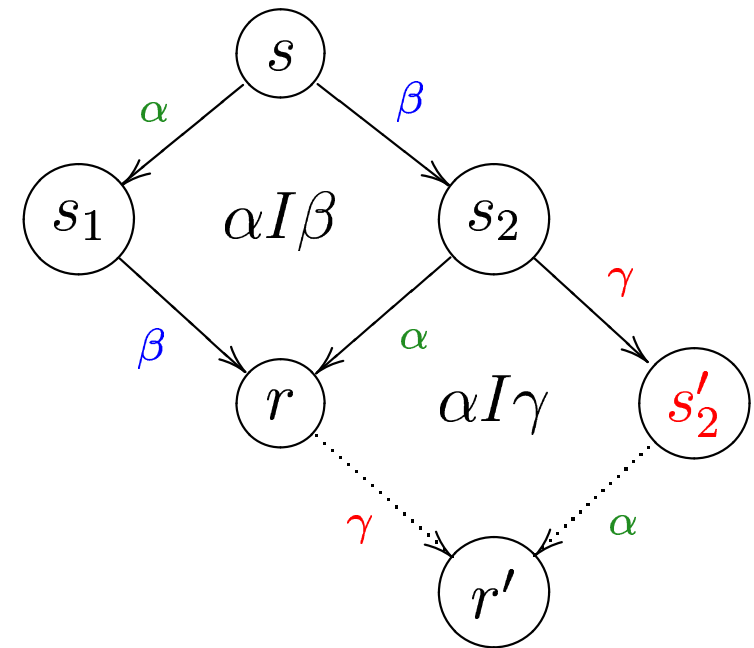
**Problem 2:** stan  $s_1$  ( $s_2$ ) może mieć następniki nieosiągalne inaczej.



np. niech  $\alpha \in \text{ample}_s, \beta \notin \text{ample}_s$

dzięki **(C1)** dla  $\beta\gamma \dots$ , wnioskujemy  $\gamma I \alpha$

**Problem 2:** stan  $s_1$  ( $s_2$ ) może mieć następniki nieosiągalne inaczej.



$\alpha$  niewidoczne, więc  $ss_1rr' \equiv ss_2s'_2$

Jak to zaimplementować?



(C1) trudny, policzymy jego przybliżenie

relacja  $D$  też będzie policzona w sposób przybliżony

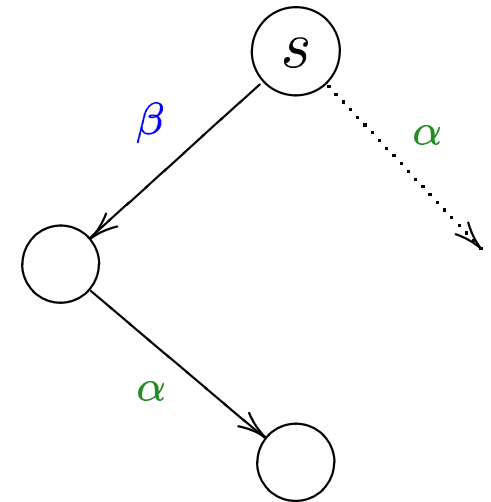
(C2) łatwy

(C3) zastąpimy przez łatwiejszy ale mocniejszy:

(C3') jeśli  $\text{ample}_s \neq \text{en}_s$  to  $\forall \alpha \in \text{ample}_s \alpha(s) \notin \text{stos}$

## Def.:

$$- \text{pre}(\alpha) \supseteq \{\beta \mid \exists s. \beta \in \text{en}_s, \alpha \notin \text{en}_s, \alpha \in \text{en}_{\beta(s)}\}$$



$$- \text{dep}(\alpha) := \{\beta \mid \alpha D \beta\}$$

**Uwaga:**  $\text{pre}(\alpha)$  i  $\text{dep}(\alpha)$  (a właściwie  $D$ ) są obliczane w sposób przybliżony

## pre( $\alpha$ ) (przybliżenie z góry)

- pre( $\alpha$ ) zawiera wszystkie operacje  $\beta$ , które zmieniają pc tak, że można wykonać  $\alpha$
- jeśli warunek umożliwiający dla  $\alpha$  zależy od zmiennych globalnych, to pre( $\alpha$ ) zawiera wszystkie operacje modyfikujące te zmienne
- jeśli  $\alpha$  to pisanie/czytanie z bufora, to pre( $\alpha$ ) zawiera wszystkie operacje czytania/pisania do tego bufora

# $\alpha D \beta$ (przybliżenie z góry)

- $\alpha$  i  $\beta$  używają tej samej zmiennej dzielonej,  
a przynajmniej jedna z nich ją modyfikuje (z góry)
- $\alpha$  i  $\beta$  są w tym samym procesie; komunikacja synchroniczna jest uważana za należącą do obydwu procesów
- $\alpha$  i  $\beta$  piszą/czytają do tego samego bufora

Ale pisanie i czytanie z tego samego bufora jest niezależne!

## Przykład:

Instrukcje niezależne od wszystkich instrukcji innych procesów:

- operacje na zmiennych lokalnych
- czytanie z bufora gdy **xr**
- pisanie do bufora gdy **xs**
- `test nempty(q)` bufora gdy **xr** dla  $q$
- `test nfull(q)` bufora gdy **xs** dla  $q$

**Def.:** (cd)

–  $pc_i(s)$  punkt sterowania (*ang. program counter*)

w procesie  $i$

–  $T_i \subseteq T$  operacje (tranzycje) procesu  $i$

–  $T_i(s) := T_i \cap en_s$   $T_i(s)$  to kandydat na  $ample_s$

–  $current_i(s) := \bigcup \{T_i(s') \mid pc_i(s') = pc_i(s)\}$

$T_i(s) \subseteq current_i(s)$

**function** ample( $s$ )

**for all**  $P_i$  **such that**  $T_i(s) \neq \emptyset$

**if**  $C1(s, i)$  **and**  $C2(T_i(s))$  **and**  $C3(s, T_i(s))$  **then**

**return**  $T_i(s)$ ;

**return**  $en_s$

**function** C1( $s, i$ )

**for all**  $P_j \neq P_i$

$\beta$  zabronione w (C1) jest

**if**  $\text{dep}(T_i(s)) \cap T_j \neq \emptyset$  **or**

w  $T_j \neq T_i$

$\text{pre}(\text{current}_i(s) \setminus T_i(s)) \cap T_j \neq \emptyset$  **then**

w  $T_i$

**return false;**

**return true**

**function** C2( $X$ ) ...

**function** C3( $s, X$ ) ...



# Redukcje cz.-p. a weryfikacja w locie

- w każdym z DFS'ów zbiór  $\text{ample}_s$  musi być ten sam
- warunek **(C3')** stosujemy do  $M \times \mathcal{A}_{\neg\phi}$  zamiast do  $M$   
czy to jest poprawne?

- malejąca widoczność

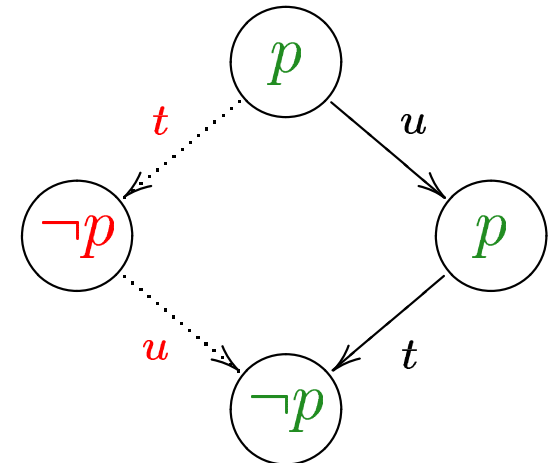
$$\phi = \mathbf{G} (p \implies \mathbf{G} q)$$

$$\neg\phi = \mathbf{F} p \wedge F\neg q$$

- $I$  zależne od  $s$

- skracanie kontrprzykładu

$\mathbf{G} p$



- BFS (tylko bezpieczeństwo)