

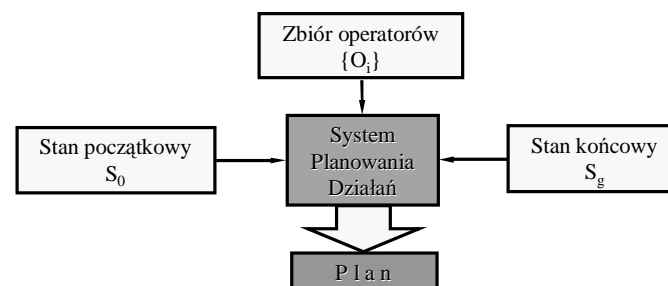
# Wprowadzenie do Sztucznej Inteligencji

Wykład 6  
Informatyka Studia Inżynierskie

©AM

## Systemy planowania działań

**Planowanie** to jest techniką rozwiązywania problemów z dziedziny AI, polegającą na określeniu ciągu akcji (operacji) jakie należy podjąć, aby przejść z zadanego stanu początkowego do stanu końcowego będącego celem.



©AM

## Reprezentacje wykorzystywane w algorytmach przeszukiwania

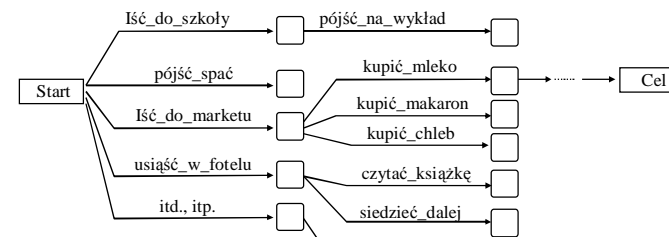
- Reprezentacja akcji - realizowana przez mechanizm generacji następników
- Reprezentacja stanów - kompletny opis stanu; akcje dokonują transformacji całych stanów; odwołanie do niej następuje w: generatorze następników, heurystycznej funkcji oceny, w funkcji wykrywania celu
- Reprezentacja celu - wykorzystywana jedynie przy oszacowaniu heurystycznym i testowaniu celu, które pełnią w algorytmie przeszukiwania wyłącznie rolę „czarnych skrzynek” - brak innych odwołań do definicji celu
- Reprezentacja rozwiązania - ciągła, nieprzerwana ścieżka od stanu początkowego do celu

©AM

## Przeszukiwanie w celu generowania planu?

- Przykład: *Zaplanować zakupy mleka, gazety i wiertarki.*

- Przestrzeń przeszukiwania:



- W systemach planowania, gdy liczba dopuszczalnych akcji jest duża, przestrzeń może być ogromna!!!

©AM

## Przeszukiwanie w celu generowania planu?

- W algorytmach przeszukiwania funkcja heurystyczna wskazuje potencjalnie najlepsze kierunki przeszukiwania
- Funkcja heurystyczna nie jest jednak nieomyślną wyrocznią!
- Skoro wybór odbywa się trochę na zasadzie „zgadywania”, każda dopuszczalna zmiana musi być przeanalizowana
- Funkcja oceny heurystycznej nie pozwala wyeliminować akcji w trakcie przeszukiwania, lecz pomaga jedynie je uporządkować
- Określona akcja jest analizowana jednak nie dlatego, że prowadzi do osiągnięcia celu, lecz dlatego, że jest dopuszczalna w danym stanie

©AM

## Planowanie: inna reprezentacja

- Reprezentacja w systemach planowania powinna:
  - dopuszczać analizę części składowych stanów, akcji i celu
  - umożliwiać dekompozycję stanów, akcji i celu
- Mechanizm planowania powinien generować plan nie koniecznie w sposób przyrostowy poprzez dodawanie akcji po kolei, poczynając od stanu inicjującego
- Większość obiektów świata, z którym planujemy jest niezależna od siebie, więc należy redukować złożoność zadania planowania poprzez jego dekompozycję na podplany
- Wniosek: tylko deklaracyjny język opisu stanów, akcji i celów pozwoli spełnić powyższe warunki

©AM

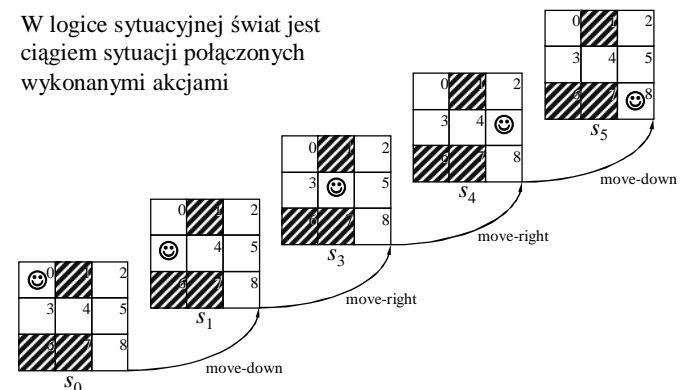
## Logika sytuacyjna

- Język formuł logiki predykatów, które opisują zmiany zachodzące w modelowanym świecie
- Świat modelowany jest jako sekwencja sytuacji, z których każda jest „stopklatką” następujących po sobie zmian
- Baza wiedzy odzwierciedla zmiany zachodzące w opisie stanu wraz z upływem czasu
- Każdej relacji i każdej własności, która może ulec zmianie przypisywany jest dodatkowy „czasowy” parametr np.:  
 $położenie(pionek, [1, 2], s_1) \wedge położenie(wieża, [3, 7], s_2)$   
 niezmiennie relacje i własności nie mają tego parametru np.:  
 $parzysta(8)$

©AM

## Przykład modelowania zmian jako sekwencji stanów

W logice sytuacyjnej świat jest ciągiem sytuacji połączonych wykonanymi akcjami



©AM

## Planowanie w logice sytuacyjnej: świat klocków

- Język logiki sytuacyjnej, służy do definiowania stanów, akcji i celów

„Świat klocków”

Trzy klocki A, B, C, które mogą być pojedynczo przenoszone oraz trzy pola, którymi nie można manipulować



- Stan świata klocków opisujemy za pomocą predykatów
- Akcje w świecie klocków definiujemy za pomocą klauzul

©AM

## Logika sytuacyjna świata klocków

- Zbiór predykatów opisu stanów i celów:
  - $on(X, Y, s_i)$  – spełniony, gdy obiekt X znajduje się na Y
  - $clear(X, s_i)$  – spełniony, gdy obiekt X jest wolny (nic na nim nie ma)
  - $moveable(X)$  – spełniony, gdy obiekt X może być przemieszczany
- Zbiór akcji:
  - Operator przemieszczania obiektów:
    - $carry(X, Y, Z)$  – akcja przeniesienia obiektu X z miejsca Y na miejsce Z
- Reprezentacja rozwiązania (propagacja zmian):
  - Funkcja:
    - $result(A, S)$  – reprezentuje stan  $S'$ , będący skutkiem wykonania akcji A stanie S, czyli  $S' = result(A, S)$

©AM

## Akcje w logice sytuacyjnej

- Akcje w logice sytuacyjnej są definiowane przez skutki ich wykonania
- Przez skutki rozumiemy te elementy opisu stanu, które zmieniają się po wykonaniu akcji np. efektem chwytania wolnego klocka (*grab*) będzie w stanie wynikowym jego trzymanie (*holding*)
 
$$\forall X, S \quad clear(X, S) \wedge moveable(X) \Rightarrow holding(X, result(grab(X), S))$$
 formuły tego typu nazywamy *aksjomatami wynikowymi (dołączania)*
- Dla pełnego opisu skutków akcji musimy również uwzględnić to co nie ulega zmianie, gdy akcja tego nie dotyczy np. fakt trzymania klocka się nie zmienia, jeśli go nie upuścimy (*drop*)
 
$$\forall A, X, S \quad holding(X, S) \wedge A \neq drop(X) \Rightarrow holding(X, result(A, S))$$
 formuły tego typu nazywamy *aksjomatami tła*

©AM

## Przykłady aksjomatów wynikowych i tła

- Aksjomaty dołączania:
 
$$on(X, Z, result(carry(X, Y, Z), S)) \Leftarrow moveable(X) \wedge X \neq Z \wedge clear(X, S) \wedge clear(Z, S) \wedge on(X, Y, S)$$

$$clear(Y, result(carry(X, Y, Z), S)) \Leftarrow moveable(X) \wedge X \neq Z \wedge clear(X, S) \wedge clear(Z, S) \wedge on(X, Y, S)$$
- Aksjomaty tła:
 
$$on(P, Q, result(carry(X, Y, Z), S)) \Leftarrow on(P, Q, S) \wedge P \neq X \wedge \dots$$

$$clear(R, result(carry(X, Y, Z), S)) \Leftarrow clear(R, S) \wedge R \neq Z \wedge \dots$$

$$\neg on(X, Y, result(carry(X, Y, Z), S)) \Leftarrow on(X, Y, S) \wedge \dots$$

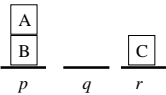
$$\neg clear(Z, result(carry(X, Y, Z), S)) \Leftarrow clear(Z, S) \wedge \dots$$

©AM

## Przykład zadania planowania w logice sytuacyjnej

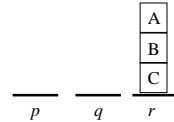
**Stan początkowy:**

$on(A, B, s_0) \wedge$   
 $on(B, p, s_0) \wedge$   
 $on(C, r, s_0) \wedge$   
 $clear(A, s_0) \wedge$   
 $clear(q, s_0) \wedge$   
 $clear(C, s_0)$



**Stan końcowy (cel):**

$\exists s \text{ on}(A, B, s) \wedge$   
 $on(B, C, s) \wedge$   
 $on(C, r, s) \wedge$   
 $clear(A, s) \wedge$   
 $clear(p, s) \wedge$   
 $clear(q, s)$



**Asercje niezależne od stanu:**

$moveable(A) \wedge$   
 $moveable(B) \wedge$   
 $moveable(C)$

**Akcje:**

$on(X, Z, result(carry(X, Y, Z, S))) \Leftarrow moveable(X) \wedge X \neq Z \wedge$   
 $clear(X, S) \wedge clear(Z, S) \wedge on(X, Y, S)$   
 $clear(Y, result(carry(X, Y, Z, S))) \Leftarrow moveable(X) \wedge X \neq Z \wedge$   
 $clear(X, S) \wedge clear(Z, S) \wedge on(X, Y, S)$   
 plus aksjomaty tła

©AM

## Problem tła (ang. frame problem)

- Problem: Jak sobie poradzić z faktem, że zdecydowana większość zdań, które były prawdziwe w poprzednim stanie pozostaje prawdziwa po wykonaniu jakiejś akcji?
- Rozwiązanie: aksjomaty tła, ale...
- Im większa liczba predykatów opisu stanu, tym większa liczba koniecznych do zdefiniowania aksjomatów tła
- Podobnie, im więcej akcji tym bardziej złożone aksjomaty tła
- Wniosek: Jedynym wyjściem jest zmiana języka opisu i sposobu reprezentacji

©AM

## Algorytm planowania w logice sytuacyjnej

- Ze względu na przyjętą reprezentację każda metoda wnioskowania stosowana w rachunku predykatów może być użyta do znalezienia planu w logice sytuacyjnej
- Zastosowanie takiego rozwiązania w realnych zadaniach planowania jest jednak niemożliwe (obliczeniowo nieefektywne) ze względu na ogromną liczbę faktów jakie trzeba przetwarzać za pomocą aksjomatów tła
- Dodatkowo, procedury dowodowe nie sterowane celem nie gwarantują żadnych własności planu poza samym faktem jego znalezienia – plan taki może być nadmiarowy np. zawierać wzajemnie znoszące się akcje  $[A_1, A_2, \dots, A, A^{-1}, \dots, A_k]$  gdzie  $s = result(A^{-1}, result(A, s))$

©AM

## Rodzaje mechanizmów generacji planów

- Planowanie w przód od stanu początkowego do stanu końcowego - propagacja stanów w przód (progresja)
- Planowanie w tył od stanu końcowego do stanu początkowego - propagacja stanów wstecz (regresja)

Systemy planowania w przód mają znaczenie tylko teoretyczne. Większość praktycznych systemów planowania, to systemy planowania wstecz.

©AM

## Planowanie progresywne: „generuj i testuj”

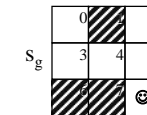
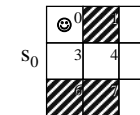
1. Pod stan aktualny podstaw stan początkowy:  
 $s_{akt} \leftarrow s_0$
2. sprawdź, czy  $s_g \subseteq s_{akt}$ :  
jeżeli tak, to koniec(sukces – plan gotowy)
3. utwórz zbiór wszystkich  $L_O$  wszystkich operatorów  $O_i$ , dla których spełnione są warunki stosowalności w stanie  $s_{akt}$ :
4. wybierz jeden operator!  $O_i \in L_O$ , wygeneruj próby stan  $s_{i+1}$ , będący efektem zastosowania tego operatora; jeżeli  $L_O = \emptyset$ , to dokonaj nawrotu do poprzedniego stanu; jeżeli brak alternatyw do nawrotów, to koniec(porażka – brak planu)
5. a. sprawdź, czy w stanie  $s_{i+1}$  pojawiły się sprzeczne podcele; jeżeli tak, to powrót do kroku 4  
b. sprawdź, czy w stanie  $s_{i+1}$  pojawił się cykl?; jeżeli tak, to nawrót do stanu poprzedniego<sup>3</sup>  $s_{akt}$  i przejście do kroku 4
6. podstaw:  $s_{akt} \leftarrow s_{i+1}$  i przejdź do kroku 2

UWAGI: 1-trzeba zastosować heurystyczny wybór np. na podstawie różnic pomiędzy stanami;  
2-cykl oznacza ponowne wystąpienie tego samego stanu;  
3-istnieje wiele sposobów eliminacji cykli!

©AM

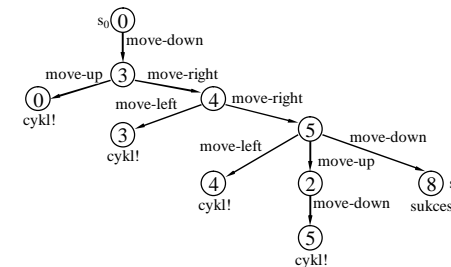
## Planowanie progresywne: „generuj i testuj”

Przykład



$O_i$ : move-up(⊙)  
move-down(⊙)  
move-left(⊙)  
move-right(⊙)

Planowanie:



©AM

## Planowanie progresywne: wady

- Proces generacji planu nie ukierunkowany stanem docelowym – groźba kombinatorycznej eksplozji stanów
- Ściśle zdeterminowany porządek generacji planu – zawsze od stanu początkowego do stanu docelowego
- Niska wydajność metody generacji planów – możliwe zastosowanie tylko dla małych przestrzeni stanów
- Efektywność silnie uzależniona od stopnia interakcji między operatorami – im głębsze niejawne zależności, tym większe wymagania zasobowe algorytmu planowania

©AM

## Systemy planowania działań: metody

- Wykorzystywane metody
  - Planowanie liniowe
  - Dekompozycja problemu (częściowa lub całkowita)
  - Regresja operatorów
  - Strategia „least-commitment”
  - Planowanie nieliniowe
  - Planowanie hierarchiczne

©AM

## Planowanie wstecz: *system STRIPS*

- Planowanie ukierunkowane stanem docelowym z wykorzystaniem stosu celów
- Reprezentacja stanów i akcji oparta na rachunku predykatów
- Specyficzna dziedzina planowania – świat klocków

©AM

## STRIPS: *świat klocków*

Założenia świata klocków:

- powierzchnia/płaszczyzna/podłoże, na którym umieszczamy klocki jest gładka i nieograniczona
- wszystkie klocki mają takie same rozmiary
- klocki mogą być umieszczone jeden na drugim
- klocki mogą tworzyć stosy
- położenie horyzontalne klocków jest nieistotne, liczy się ich wertykalne położenie względem siebie
- manipulujemy klockami tylko za pomocą ramienia robota
- w danej chwili w ramieniu robota może znajdować się tylko jeden klocek

©AM

## STRIPS: *zbiór operatorów*

**STACK(x,y):** umieszczenie klocka x na klocku y; w ramieniu robota musi znajdować się klocek x, a na klocku y nie może znajdować się żaden klocek

**UNSTACK(x,y):** zdjęcie klocka x z klocka y; ramię robota musi być puste/wolne a na klocku x nie może znajdować się inny klocek

**PICKUP(x):** podniesienie klocka x z podłoża; ramię robota musi być puste/wolne a na klocku x nie może znajdować się inny klocek

**PUTDOWN(x):** umieszczenie klocka x na podłożu; w ramieniu robota musi znajdować się klocek x

©AM

## STRIPS: *zbiór predykatów*

**ON(x,y)** spełniony, gdy klocek x znajduje się na klocku y

**ONTABLE(x)** spełniony, gdy klocek x znajduje się bezpośrednio na podłożu

**CLEAR(x)** spełniony, gdy powierzchnia klocka x jest pusta tzn. nie znajduje się na nim żaden inny klocek

**HOLDING(x)** spełniony, gdy w ramieniu robota znajduje się klocek x

**ARMEMPTY** spełniony, gdy ramię robota jest puste/wolne

Uniwersalne reguły rządzące światem klocków są reprezentowane z wykorzystaniem tych predykatów jako aksjomaty zapisane w postaci formuł rachunku predykatów, np.:

$[\exists x \text{ HOLDING}(x)] \Rightarrow \neg \text{ARMEMPTY}$

$\forall x \text{ ONTABLE}(x) \Rightarrow \neg \exists y \text{ ON}(x,y)$

$\forall x [\neg \exists y \text{ ON}(y,x)] \Rightarrow \text{CLEAR}(x)$

©AM

## STRIPS: reprezentacja operatorów

*Problem tła* (ang. frame problem) w systemie STRIPS rozwiązano stosując odpowiednią formę reprezentacji operatorów.

Każdy operatora zawiera:

- listę predykatów, które muszą być prawdziwe, aby jego użycie było możliwe - sekcja PRECONDITION - warunki stosowalności operatora
- listę predykatów, które staną się prawdziwe po jego wykonaniu - sekcja ADD
- listę predykatów, które przestaną być prawdziwe po jego wykonaniu - sekcja DELETE

Wszystkie predykaty, które *nie znalazły* się na listach ADD i DELETE pozostają niezmienione po wykonaniu operatora. Listy ADD oraz DELETE opisują łącznie efekty użycia operatora.

©AM

## STRIPS: definicja formalna operatorów

### STACK(x,y):

PRECONDITION:  $CLEAR(y) \wedge HOLDING(x)$   
DELETE:  $CLEAR(y) \wedge HOLDING(x)$   
ADD:  $ARMEMPTY \wedge ON(x,y)$

### UNSTACK(x,y):

PRECONDITION:  $ON(x,y) \wedge CLEAR(x) \wedge ARMEMPTY$   
DELETE:  $ON(x,y) \wedge ARMEMPTY$   
ADD:  $HOLDING(x) \wedge CLEAR(y)$

### PICKUP(x):

PRECONDITION:  $CLEAR(x) \wedge ONTABLE(x) \wedge ARMEMPTY$   
DELETE:  $ONTABLE(x) \wedge ARMEMPTY$   
ADD:  $HOLDING(x)$

### PUTDOWN(x):

PRECONDITION:  $HOLDING(x)$   
DELETE:  $HOLDING(x)$   
ADD:  $ONTABLE(x) \wedge ARMEMPTY$

©AM

## STRIPS: algorytm planowania wstecz

Stos celów - podstawowa struktura danych generatora planu, zawierająca zarówno cele, jak i operatory służące do ich osiągnięcia

Przebieg planowania:

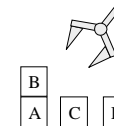
- wykrywanie różnic między stanem aktualnym a celem – umieszczenie podcelów niespełnionych w akt. stanie na stosie
- wybór operatora redukującego różnicę
- zastąpienie podcelu wybranym operatorem na stosie
- umieszczenie na stosie warunków stosowalności wybranego operatora - nowy (złożony) cel
- rozkład warunku stosowalności na podcele - nowe cele jednostkowe (proste) na stosie

©AM

## STRIPS: przykład planowania

### Stan początkowy:

$ON(B,A) \wedge$   
 $ONTABLE(A) \wedge$   
 $ONTABLE(C) \wedge$   
 $ONTABLE(D) \wedge$   
 $ARMEMPTY$



### Stan końcowy (docelowy):

$ON(C,A) \wedge$   
 $ON(B,D) \wedge$   
 $ONTABLE(A) \wedge$   
 $ONTABLE(D)$



©AM

55

## STRIPS: przykład planowania

Stos celów: Stan aktualny:

Umieszczamy stan docelowy na stosie

$ON(C,A) \wedge ON(B,D) \wedge ONTABLE(A) \wedge ONTABLE(D)$

B  
A C D

©AM

56

## STRIPS: przykład planowania

Stos celów: Stan docelowy:

Różnica między stanem aktualnym a celem

$ON(C,A)$   
 $ON(B,D)$   
 $ON(C,A) \wedge ON(B,D) \wedge ONTABLE(A) \wedge ONTABLE(D)$

C B  
A D

B  
A C D

©AM

57

## STRIPS: przykład planowania

Stos celów: Stan aktualny:

Operator spełniający cel ze szczytu stosu

STACK(C,A)  
ON(B,D)  
 $ON(C,A) \wedge ON(B,D) \wedge ONTABLE(A) \wedge ONTABLE(D)$

B  
A C D

©AM

58

## STRIPS: przykład planowania

Stos celów: Stan aktualny:

Warunek stosowalności operatora

CLEAR(A)  $\wedge$  HOLDING(C)  
STACK(C,A)  
ON(B,D)  
 $ON(C,A) \wedge ON(B,D) \wedge ONTABLE(A) \wedge ONTABLE(D)$

B  
A C D

©AM



59

## STRIPS: przykład planowania

Stos celów:      Stan aktualny:

CLEAR(A)
HOLDING(C)
CLEAR(A) $\wedge$ HOLDING(C)
STACK(C,A)
ON(B,D)
ON(C,A) $\wedge$ ON(B,D) $\wedge$ ONTABLE(A) $\wedge$ ONTABLE(D)

Rozbicie warunku na podwarunki: kolejność wynika z wiedzy heurystycznej (stan HOLDING łatwiej zniweczyć)

©AM

60

## STRIPS: przykład planowania

Stos celów:      Stan aktualny:

UNSTACK(B,A)
HOLDING(C)
CLEAR(A) $\wedge$ HOLDING(C)
STACK(C,A)
ON(B,D)
ON(C,A) $\wedge$ ON(B,D) $\wedge$ ONTABLE(A) $\wedge$ ONTABLE(D)

Operator spełniający cel ze szczytu stosu

©AM

61

## STRIPS: przykład planowania

Stos celów:      Stan aktualny:

ON(B,A) $\wedge$ CLEAR(B) $\wedge$ ARMEPMTY
UNSTACK(B,A)
HOLDING(C)
CLEAR(A) $\wedge$ HOLDING(C)
STACK(C,A)
ON(B,D)
ON(C,A) $\wedge$ ON(B,D) $\wedge$ ONTABLE(A) $\wedge$ ONTABLE(D)

Warunek stosowalności operatora

©AM

62

## STRIPS: przykład planowania

Stos celów:      Stan aktualny:

ON(B,A)
CLEAR(B)
ARMEPMTY
ON(B,A) $\wedge$ CLEAR(B) $\wedge$ ARMEPMTY
UNSTACK(B,A)
HOLDING(C)
CLEAR(A) $\wedge$ HOLDING(C)
STACK(C,A)
ON(B,D)
ON(C,A) $\wedge$ ON(B,D) $\wedge$ ONTABLE(A) $\wedge$ ONTABLE(D)

Rozbicie warunku na podwarunki: kolejność heurystyczna

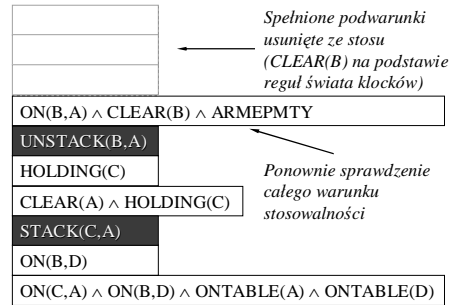
©AM

## STRIPS: przykład planowania

63

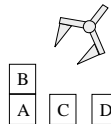
Stos celów:

Stan aktualny:



Spełnione podwarunki  
usunięte ze stosu  
(CLEAR(B) na podstawie  
reguł świata klocków)

Ponownie sprawdzenie  
całego warunku  
stosowalności



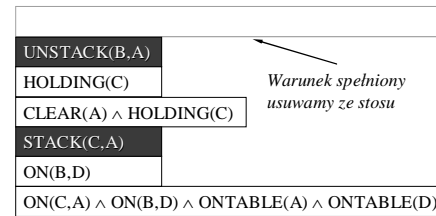
©AM

## STRIPS: przykład planowania

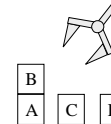
64

Stos celów:

Stan aktualny:



Warunek spełniony  
usuujemy ze stosu



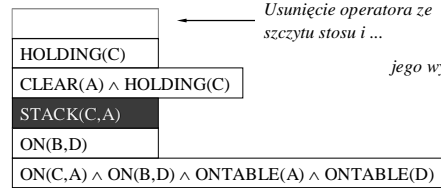
©AM

## STRIPS: przykład planowania

65

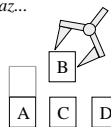
Stos celów:

Stan aktualny:



Usunięcie operatora ze  
szczytu stosu i ...

jego wykonanie oraz...



Plan aktualny:

UNSTACK(B,A)

dodanie operatora do  
planu

HOLDING(B)  $\wedge$   
ONTABLE(A)  $\wedge$   
ONTABLE(C)  $\wedge$   
ONTABLE(D)  $\wedge$   
CLEAR(A)

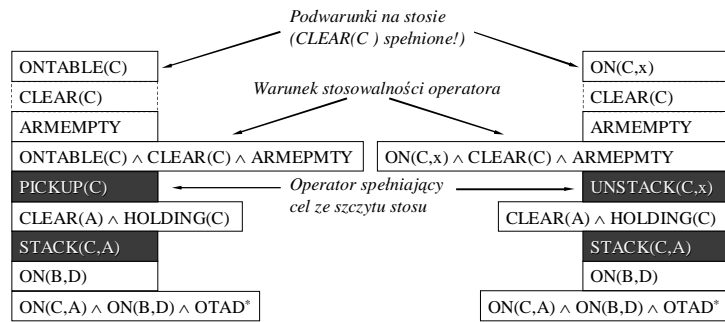
©AM

## STRIPS: przykład planowania

66

Stos celów:

Alternatywny stos celów:



OTAD\* = ONTABLE(A)  $\wedge$  ONTABLE(D)

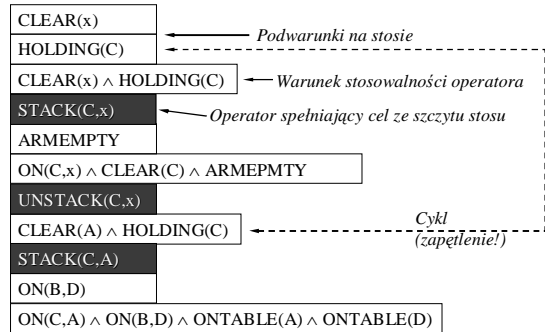
©AM

## STRIPS: przykład planowania

67

Alternatywny stos celów:

Stan aktualny:



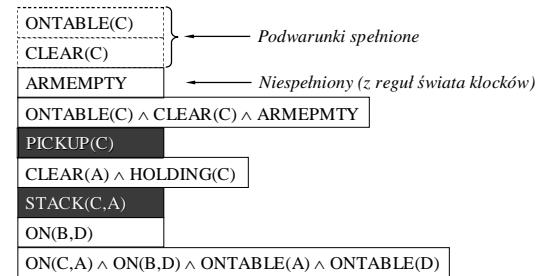
©AM

## STRIPS: przykład planowania

68

Stos celów:

Stan aktualny:



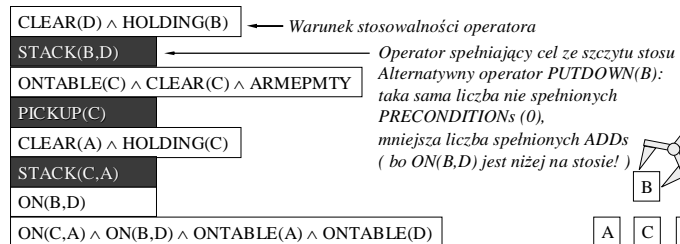
©AM

## STRIPS: przykład planowania

69

Stos celów:

Stan aktualny:



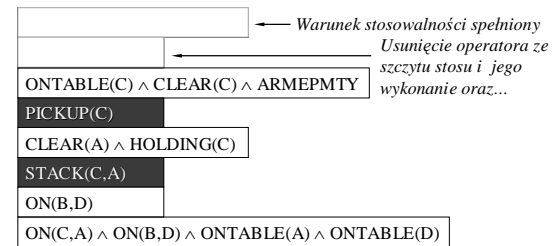
©AM

## STRIPS: przykład planowania

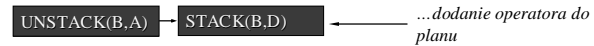
70

Stos celów:

Stan aktualny:



Plan aktualny:



©AM

71

## STRIPS: przykład planowania

Stos celów:      Stan aktualny:

← Warunek stosowalności spełniony

← Usunięcie operatora ze szczytu stosu i jego wykonanie oraz...

CLEAR(A) ∧ HOLDING(C)

STACK(C,A)

ON(B,D)

ON(C,A) ∧ ON(B,D) ∧ ONTABLE(A) ∧ ONTABLE(D)

---

Plan aktualny:

UNSTACK(B,A) → STACK(B,D) → PICKUP(C) ← ...dodanie operatora do planu

©AM

72

## STRIPS: przykład planowania

Stos celów:      Stan aktualny:

← Warunek stosowalności spełniony

← Usunięcie operatora ze szczytu stosu i jego wykonanie oraz ...

ON(B,D)

ON(C,A) ∧ ON(B,D) ∧ ONTABLE(A) ∧ ONTABLE(D)

---

Plan aktualny:

UNSTACK(B,A) → STACK(B,D) → PICKUP(C) → STACK(C,A) ← ...dodanie operatora do planu

©AM

73

## STRIPS: przykład planowania

Stos celów:      Stan aktualny:

← Warunek spełniony

← Cel główny spełniony

↑  
Stos pusty – KONIEC planowania!

---

Plan aktualny:

UNSTACK(B,A) → STACK(B,D) → PICKUP(C) → STACK(C,A)

©AM

74

## STRIPS: koniec przykładu

Stan początkowy:      Stan końcowy (docelowy):

ON(B,A) ∧  
ONTABLE(A) ∧  
ONTABLE(C) ∧  
ONTABLE(D) ∧  
ARMEPMTY

---

ON(C,A) ∧  
ON(B,D) ∧  
ONTABLE(A) ∧  
ONTABLE(D)

---

Plan:

UNSTACK(B,A) → STACK(B,D) → PICKUP(C) → STACK(C,A)

©AM

## STRIPS: heurystyka

75

Wiedza heurystyczna:

- kolejność umieszczania różnic (warunków) na stosie
- wybór operatora redukującego różnicę

Kolejność różnic/warunków:

- wiedza heurystyczna – wiedza o trudności w osiągnięciu warunków (ten, który łatwo zniweczyć spełniamy później)
- zapis wiedzy heurystycznej – odpowiednia kolejność warunków w sekcji PRECONDITIONs operatora

Wybór operatora:

- ten, który szybciej przybliży stan aktualny do stanu docelowego (ma mniejszą liczę niespełnionych warunków stosowalności)
- drugie kryterium: dodatkowo spełnia inny cel niż na stosie (efekt uboczny)

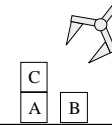
©AM

## STRIPS: anomalia Sussman'a

76

Stan początkowy:

ON(C,A)  $\wedge$   
 ONTABLE(A)  $\wedge$   
 ONTABLE(B)  $\wedge$   
 ARMEPMTY



Stan końcowy (docelowy):

ON(A,B)  $\wedge$   
 ON(B,C)  $\wedge$   
 ONTABLE(C)



©AM

## STRIPS: anomalia Sussman'a

77

Stos celów:

Alternatywny stos celów:

ON(A,B)  
 ON(B,C)  
 ONTABLE(C)

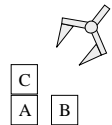
ON(B,C)  
 ON(A,B)  
 ONTABLE(C)

ON(A,B)  $\wedge$  ON(B,C)  $\wedge$  ONTABLE(C)

ON(A,B)  $\wedge$  ON(B,C)  $\wedge$  ONTABLE(C)

Stan aktualny:

ON(C,A)  $\wedge$   
 ONTABLE(A)  $\wedge$   
 ONTABLE(B)  $\wedge$   
 ARMEPMTY



©AM

## STRIPS: anomalia Sussman'a

78

Stos celów:

Stan aktualny:

CLEAR(C)  
 ARMEPMTY  
 ON(C,A)

CLEAR(C)  $\wedge$  ARMEPMTY  $\wedge$  ON(C,A)

UNSTACK(C,A)

ARMEPMTY

CLEAR(A)  $\wedge$  ARMEPMTY  $\wedge$  ONTABLE(A)

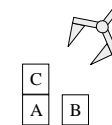
PICKUP(A)

CLEAR(B)  $\wedge$  HOLDING(A)

STACK(A,B)

ON(B,C)

ON(A,B)  $\wedge$  ON(B,C)  $\wedge$  ONTABLE(C)



©AM

79

## STRIPS: *anomalia Sussman'a*

**Stos celów:**

ARMEMPTY

CLEAR(A)  $\wedge$  ARMEMPTY  $\wedge$  ONTABLE(A)

PICKUP(A)

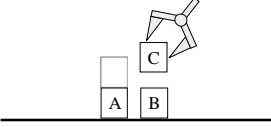
CLEAR(B)  $\wedge$  HOLDING(A)

STACK(A,B)

ON(B,C)

ON(A,B)  $\wedge$  ON(B,C)  $\wedge$  ONTABLE(C)

**Stan aktualny:**



**Plan aktualny:**

UNSTACK(C,A)

©AM

80

## STRIPS: *anomalia Sussman'a*

**Stos celów:**

HOLDING(C) ← Warunek stosowalności

PUTDOWN(C) ← Wybrany, bo spełnia ONTABLE(C) poniżej

CLEAR(A)  $\wedge$  ARMEMPTY  $\wedge$  ONTABLE(A)

PICKUP(A)

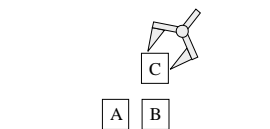
CLEAR(B)  $\wedge$  HOLDING(A)

STACK(A,B)

ON(B,C)

ON(A,B)  $\wedge$  ON(B,C)  $\wedge$  ONTABLE(C)

**Stan aktualny:**



©AM

81

## STRIPS: *anomalia Sussman'a*

**Stos celów:**

CLEAR(A)  $\wedge$  ARMEMPTY  $\wedge$  ONTABLE(A)

PICKUP(A)

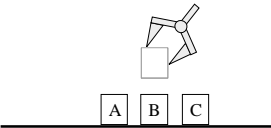
CLEAR(B)  $\wedge$  HOLDING(A)

STACK(A,B)

ON(B,C)

ON(A,B)  $\wedge$  ON(B,C)  $\wedge$  ONTABLE(C)

**Stan aktualny:**



**Plan aktualny:**

UNSTACK(C,A)

↓

PUTDOWN(C)

©AM

82

## STRIPS: *anomalia Sussman'a*

**Stos celów:**

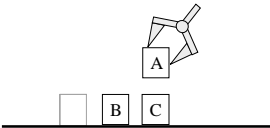
CLEAR(B)  $\wedge$  HOLDING(A)

STACK(A,B)

ON(B,C)

ON(A,B)  $\wedge$  ON(B,C)  $\wedge$  ONTABLE(C)

**Stan aktualny:**



**Plan aktualny:**

UNSTACK(C,A)

↓

PUTDOWN(C)

↓

PICKUP(A)

©AM



87

### STRIPS: *anomalia Sussman'a*

Stos celów:	Plan aktualny:
HOLDING(A)	UNSTACK(C,A)
PUTDOWN(A) ← Nie STACK(A,B) bo byłby cykl	↓
CLEAR(B) ∧ ARMEMPTY ∧ ONTABLE(B)	PUTDOWN(C)
PICKUP(B)	↓
CLEAR(C) ∧ HOLDING(B)	PICKUP(A)
STACK(B,C)	↓
ON(A,B) ∧ ON(B,C) ∧ ONTABLE(C)	STACK(A,B)
	↓
	UNSTACK(A,B)
Stan aktualny:	

©AM

88

### STRIPS: *anomalia Sussman'a*

Stos celów:	Plan aktualny:
	UNSTACK(C,A)
	↓
CLEAR(B) ∧ ARMEMPTY ∧ ONTABLE(B)	PUTDOWN(C)
PICKUP(B)	↓
CLEAR(C) ∧ HOLDING(B)	PICKUP(A)
STACK(B,C)	↓
ON(A,B) ∧ ON(B,C) ∧ ONTABLE(C)	STACK(A,B)
	↓
	UNSTACK(A,B)
	↓
	PUTDOWN(A)
Stan aktualny:	

©AM

89

### STRIPS: *anomalia Sussman'a*

Stos celów:	Plan aktualny:
	UNSTACK(C,A)
	↓
	PUTDOWN(C)
	↓
	PICKUP(A)
CLEAR(C) ∧ HOLDING(B)	↓
STACK(B,C)	STACK(A,B)
ON(A,B) ∧ ON(B,C) ∧ ONTABLE(C)	↓
	UNSTACK(A,B)
	↓
	PUTDOWN(A)
	↓
	PICKUP(B)
Stan aktualny:	

©AM

90

### STRIPS: *anomalia Sussman'a*

Stos celów:	Plan aktualny:
	UNSTACK(C,A)
	↓
	PUTDOWN(C)
	↓
	PICKUP(A)
	↓
	STACK(A,B)
ON(A,B) ∧ ON(B,C) ∧ ONTABLE(C)	↓
	UNSTACK(A,B)
	↓
	PUTDOWN(A)
	↓
	PICKUP(B)
	↓
	STACK(B,C)
Stan aktualny:	

©AM



91

### STRIPS: *anomia Sussman'a*

**Stos celów:**

$CLEAR(A) \wedge ARMEMPTY \wedge ONTABLE(A)$

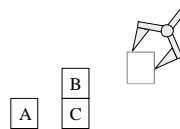
PICKUP(A)

$CLEAR(B) \wedge HOLDING(A)$

STACK(A,B) ← *Ponownie nie jest spełnione!*

$ON(A,B) \wedge ON(B,C) \wedge ONTABLE(C)$

**Stan aktualny:**



**Plan aktualny:**

UNSTACK(C,A)

↓

PUTDOWN(C)

↓

PICKUP(A)

↓

STACK(A,B)

↓

UNSTACK(A,B)

↓

PUTDOWN(A)

↓

PICKUP(B)

↓

STACK(B,C)

©AM

92

### STRIPS: *anomia Sussman'a*

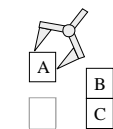
**Stos celów:**

$CLEAR(B) \wedge HOLDING(A)$

STACK(A,B)

$ON(A,B) \wedge ON(B,C) \wedge ONTABLE(C)$

**Stan aktualny:**



**Plan aktualny:**

UNSTACK(C,A)

↓

PUTDOWN(C)

↓

PICKUP(A)

↓

STACK(A,B)

↓

UNSTACK(A,B)

↓

PUTDOWN(A)

↓

PICKUP(B)

↓

STACK(B,C)

↓

PICKUP(A)

©AM

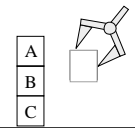
93

### STRIPS: *anomia Sussman'a*

**Stos celów:**

*Stos pusty – koniec planowania!*

**Stan aktualny:**



**Plan aktualny:**

UNSTACK(C,A)

↓

PUTDOWN(C)

↓

PICKUP(A)

↓

STACK(A,B)

↓

UNSTACK(A,B)

↓

PUTDOWN(A)

↓

PICKUP(B)

↓

STACK(B,C)

↓

PICKUP(A)

↓

STACK(A,B)

©AM

95

### STRIPS: *anomia Sussman'a*

Otrzymany plan nie jest optymalny:

- niektóre operacje zaraz po wykonaniu zostaną wycofane (bezpośredni następnik w planie to operacja odwrotna)
- takie dopełniające się pary operacji można wykryć i usunąć z planu w drodze analizy końcowego planu, ale nie we wszystkich dziedzinach możliwe jest określenie, która operacja niweluje inną;
- dodatkowo, zbędne okazało się całe przetwarzanie, które zostało wykonane w trakcie planowania dla dopełniających się par operatorów

**Plan aktualny:**

UNSTACK(C,A)

↓

PUTDOWN(C)

↓

PICKUP(A)

↓

STACK(A,B)

↓

UNSTACK(A,B)

↓

PUTDOWN(A)

↓

PICKUP(B)

↓

STACK(B,C)

↓

PICKUP(A)

↓

STACK(A,B)

©AM

## STRIPS: *anomia Sussman'a*

96

Otrzymany plan nie jest optymalny:

- niektóre operacje zaraz po wykonaniu zostaną wycofane (bezpośredni następnik w planie to operacja odwrotna)
- takie dopełniające się pary operacji można wykryć i usunąć z planu, ale nie we wszystkich dziedzinach możliwe jest określenie, która operacją niweluje inną;
- dodatkowo, zbędne okazało się całe przetwarzanie, które zostało wykonane w trakcie planowania dla takich dopełniających się par operatorów

Czy za nieoptymalny plan odpowiada użyta heurystyka, czy sam algorytm planowania liniowego?

Sprawdźmy zatem skutki innych decyzji heurystycznych!

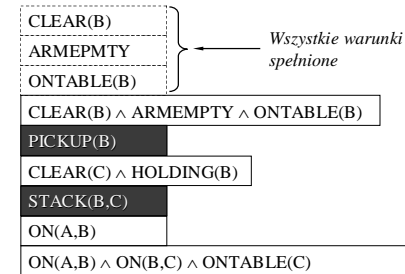
©AM

## STRIPS: *anomia Sussman'a*

97

Alternatywny stos celów:

Stan aktualny:



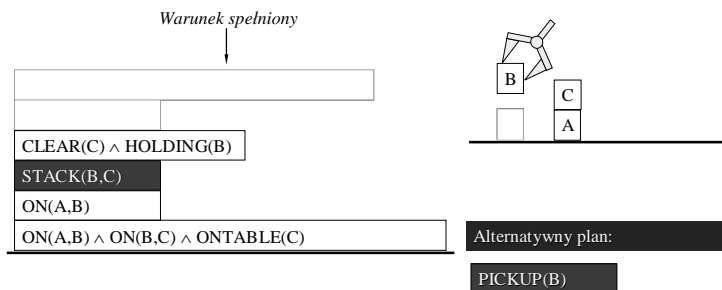
©AM

## STRIPS: *anomia Sussman'a*

98

Alternatywny stos celów:

Stan aktualny:



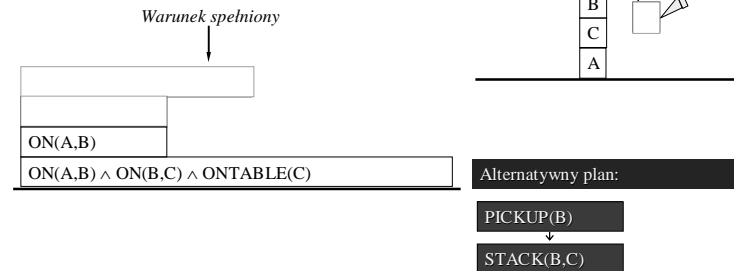
©AM

## STRIPS: *anomia Sussman'a*

99

Alternatywny stos celów:

Stan aktualny:



©AM

## STRIPS: *anomia Sussman'a*

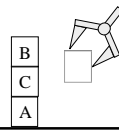
Alternatywny stos celów:

Stan aktualny:

itd.

ON(A,B)

ON(A,B)  $\wedge$  ON(B,C)  $\wedge$  ONTABLE(C)

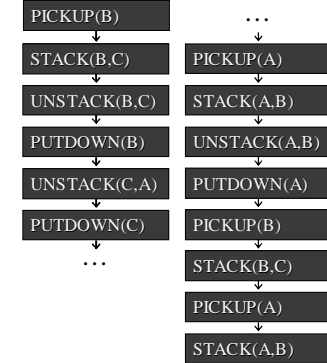


©AM

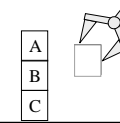
## STRIPS: *anomia Sussman'a*

Stan początkowy:

Alternatywny kompletny plan:



Stan końcowy:



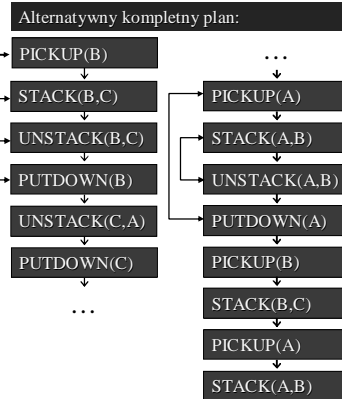
©AM

## STRIPS: *anomia Sussman'a*

Alternatywny plan również nie jest optymalny (a nawet jest gorszy!):

- jest dłuższy (o 4 operacje)
- zawiera więcej par niwelujących się operacji

Za brak optymalności planowania nie odpowiada zatem użyta heurystyka, lecz sam algorytm planowania liniowego!



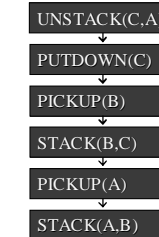
©AM

## STRIPS: *anomia Sussman'a*

Stan początkowy:

Stan końcowy:

Optymalny plan:



Nieoptymalny plan nie jest jedynie efektem użytej heurystyki, lecz wynika z ograniczeń samego algorytmu planowania liniowego - nie jest istotna kolejność osiągania celów i porządek wybierania operatorów, lecz fakt, iż metoda nie uwzględnia interakcji zachodzących pomiędzy warunkami i operatorami.

©AM

## Poszukiwanie planu: metoda regresji celów

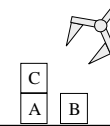
- Planowanie opiera się na *zbiorze celów*, spośród których dowolny podcel może być wybrany jako kolejny, który należy osiągnąć
- Planowanie odbywa się *wstecz* (regresywnie) – od stanu docelowego do stanu początkowego
- Wybierany ze zbioru podcel jest traktowany tak jakby był *ostatnim*, który należy osiągnąć (inaczej: zakładamy, że reszta celów jest już spełniona i szukamy tylko tych operacji, które miałyby dopełnić plan poprzez spełnienie „ostatniego” podcelu)
- Wybrany operator *poszerza zbiór podcelów* o swoje warunki stosowności

©AM

## Regresja celów: przykład (anomia Sussman'a)

Stan początkowy:

ON(C,A)  $\wedge$   
 ONTABLE(A)  $\wedge$   
 ONTABLE(B)  $\wedge$   
 ARMEPMTY



Stan końcowy (docelowy):

ON(A,B)  $\wedge$   
 ON(B,C)  $\wedge$   
 ONTABLE(C)\*

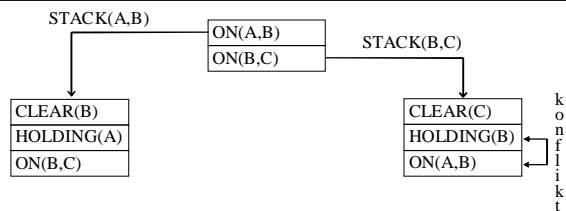


©AM

\* - dla uproszczenia rozważań podcel ONTABLE(C) nie będzie brany pod uwagę

## Regresja celów: przykład

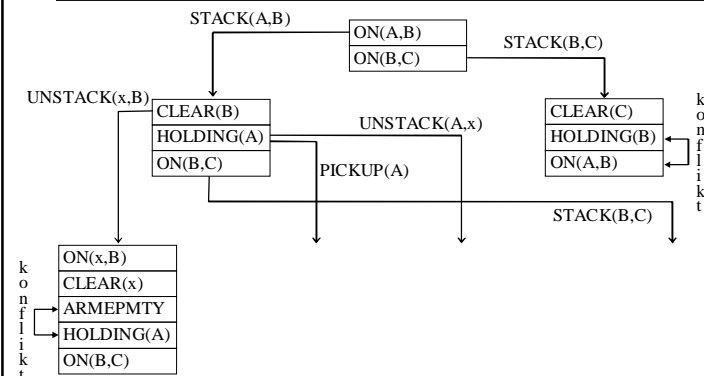
Drzewo przeszukiwania w przestrzeni zbiorów celów:



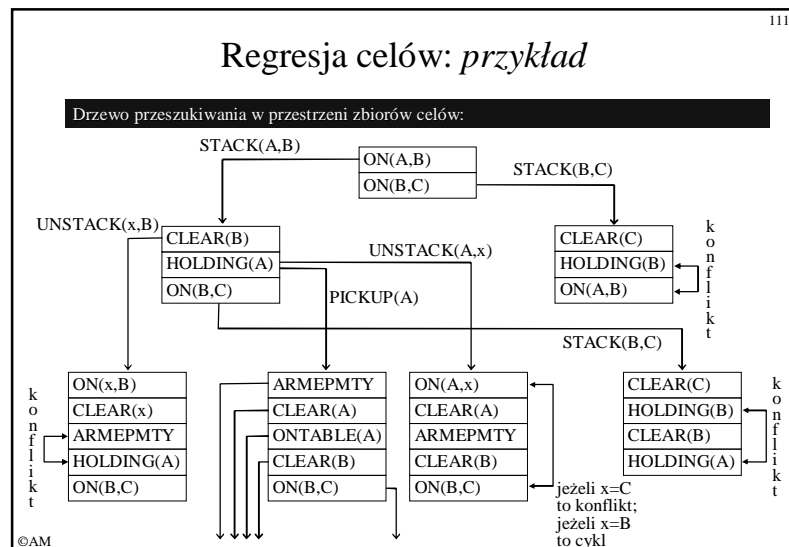
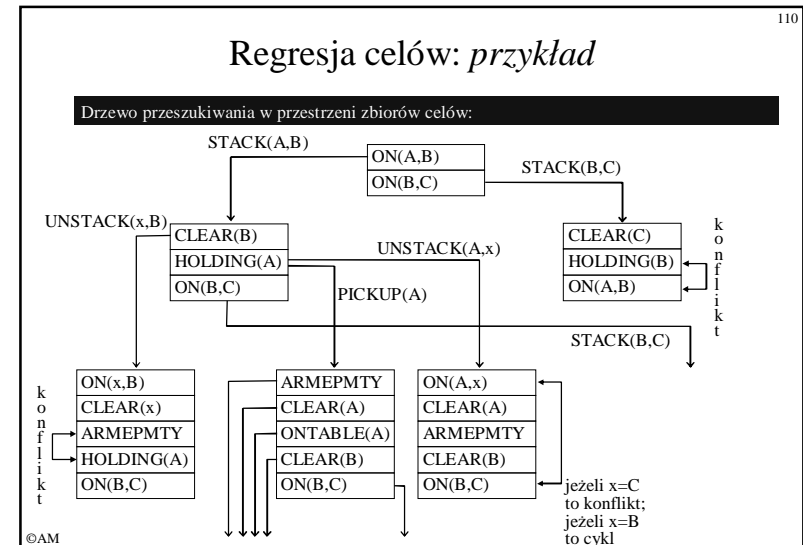
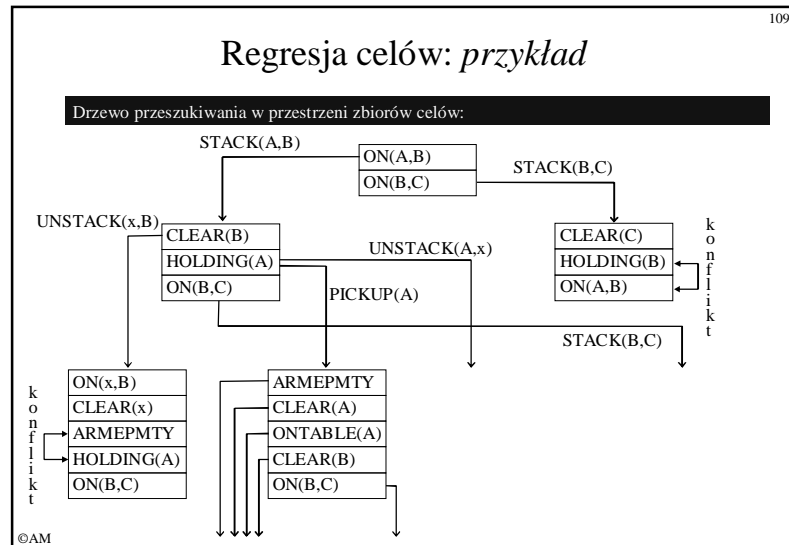
©AM

## Regresja celów: przykład

Drzewo przeszukiwania w przestrzeni zbiorów celów:



©AM



- 112
- ### Regresja celów: operacja regresji
- Planowanie wstecz opiera się na założeniu, że operator wybrany do spełnienia podcelu jest ostatnim w planie, a pozostałe podcele zarówno przed, jak i po jego wykonaniu są już spełnione
  - Założenie to jest prawdziwe tylko wtedy, gdy operatory są całkowicie niezależne; najczęściej jednak jest inaczej - wykonanie jednego operatora może zniweczyć skutki działania innego operatora
  - Wsteczne zastosowanie operatora, określane mianem *regresji operatora*, wymaga każdorazowo sprawdzenia jakie warunki muszą być spełnione, aby wykonanie operacji było możliwe i prowadziło do osiągnięcia podcelu bez naruszania innych (już spełnionych) podcelów
- ©AM

### Regresja celów: *operacja regresji*

113

- Regresja podcelu za pomocą operatora, który nie wpływa na ten podcel daje w rezultacie ten sam podcel\*, np.

$\text{REGRESSION}(\text{ON}(\text{A},\text{B}), \text{PICKUP}(\text{C})) = \text{ON}(\text{A},\text{B})$

\* - jest to najczęstszy przypadek regresji operatora

©AM

### Regresja celów: *operacja regresji*

114

- Regresja podcelu operatorem, których prowadzi do osiągnięcia tego podcelu nie wymaga spełnienia żadnych warunków (warunki stosowalności operatora są „obsługiwane” w inny sposób\*), np.

$\text{REGRESSION}(\text{ON}(\text{A},\text{B}), \text{STACK}(\text{A},\text{B})) = \text{True}$

\* - zgodnie z algorytmem dodawane są do zbioru celów

©AM

### Regresja celów: *operacja regresji*

115

- Regresja podcelu za pomocą niektórych operatorów może prowadzić do sytuacji, w której spełnienie tego celu nie będzie już więcej możliwe\*, np.

$\text{REGRESSION}(\text{ARMEPMTY}, \text{PICKUP}(\text{C})) = \text{False}$

\* - taka ścieżka poszukiwania planu oznacza brak możliwości dalszego planowania

©AM

### Regresja celów: *operacja regresji*

116

- Regresja jednego podcelu może - jako efekt uboczny - prowadzić również do osiągnięcia podcelu, do którego spełnienia bezpośrednio nie została użyta\*, np.

$\text{REGRESSION}(\text{ON}(\text{B},\text{D}), \text{STACK}(\text{B},\text{D})) = \text{True}$

ale również

$\text{REGRESSION}(\text{ARMEPMTY}, \text{STACK}(\text{B},\text{D})) = \text{True}$

\* - podcele o wartości True mogą zostać usunięte ze zbioru celów jako na pewno spełnione

©AM

## Regresja celów: warunki zakończenia planowania

117

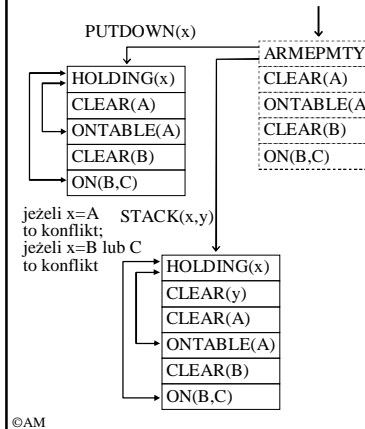
- Proces planowania odbywa się wstecz - nie wiadomo jakie operacje zostały (a w zasadzie: zostaną) wykonane przed aktualnym operatorem oraz jakie były ich skutki poza tym, że doprowadziły do spełnienia wszystkich oprócz jednego podcelu ze zbioru celów
- Nie są znane skutki uboczne wykonania tych wcześniejszych operacji i ich ewentualny wpływ na inne podcele - żaden podcel nie może być zignorowany zanim nie ulegnie regresji do warunku *True*
- Proces planowania może zakończyć się, gdy mamy pewność, że żadne warunki nie ulegną już zmianie - wszystkie podcele są jednocześnie spełnione w stanie początkowym i nie ma potrzeby stosowania jakichkolwiek operatorów poprzedzających ciąg operacji dopiero co znaleziony

©AM

## Regresja celów: przykład

119

Drzewo poszukiwania planu w przestrzeni zbiorów celów:

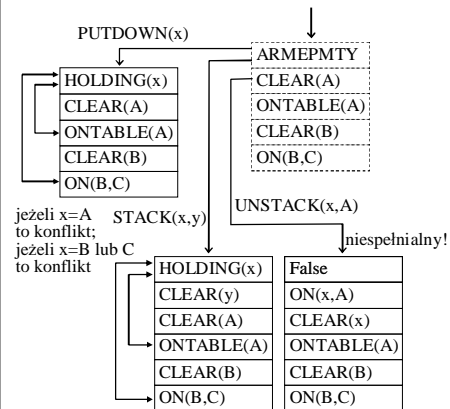


©AM

## Regresja celów: przykład

120

Drzewo poszukiwania planu w przestrzeni zbiorów celów:

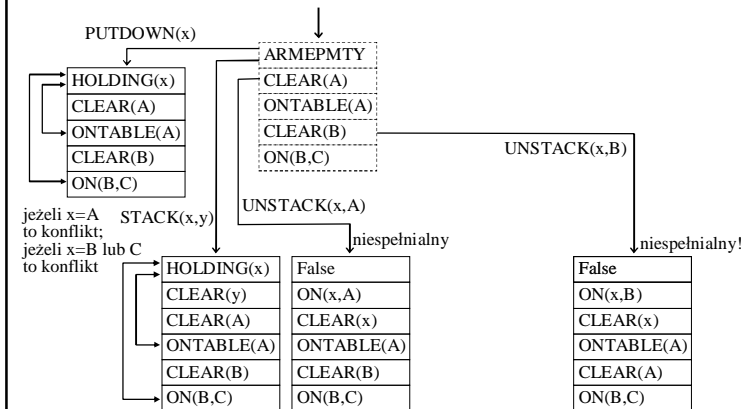


©AM

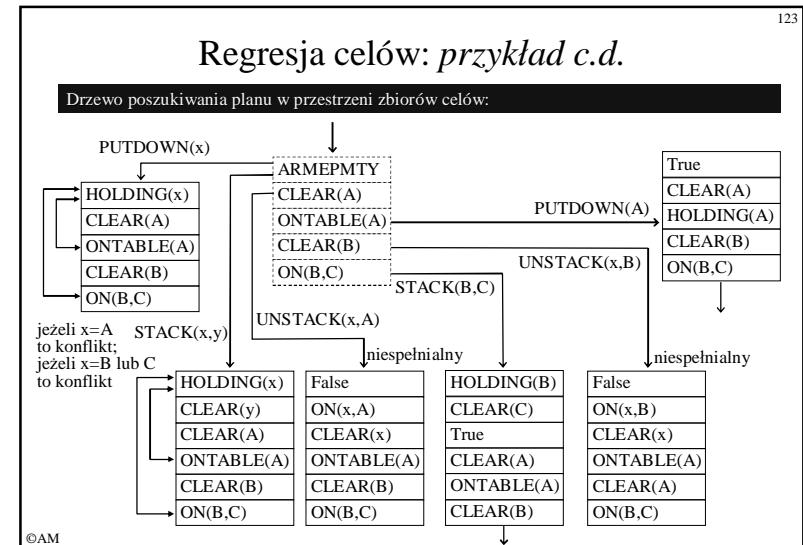
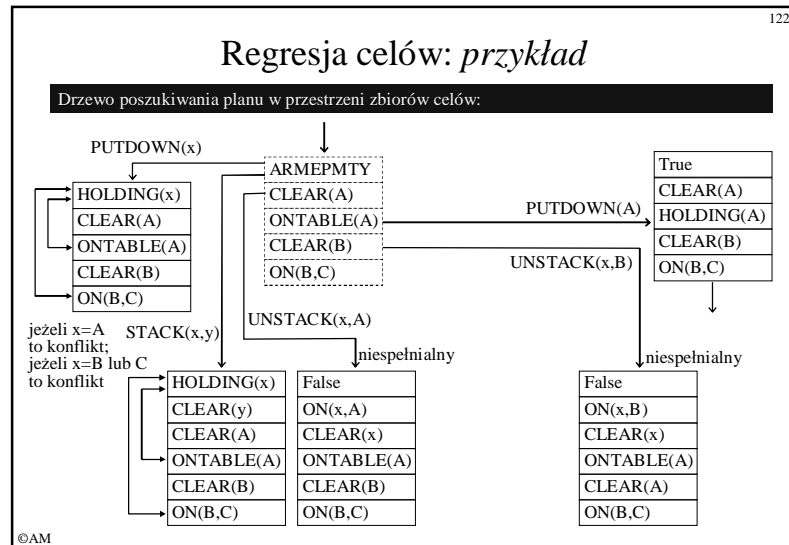
## Regresja celów: przykład

121

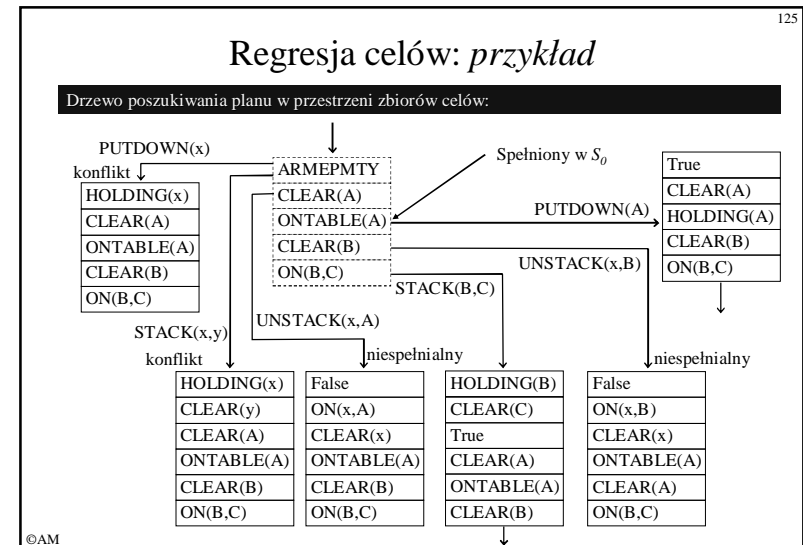
Drzewo poszukiwania planu w przestrzeni zbiorów celów:



©AM



- 124
- ### Regresja celów: heurystyki
- Warunkiem zatrzymania procesu planowania jest osiągnięcie stanu początkowego (wnioskowanie wstecz) - wiedza o tym stanie jest zatem kluczowa dla podejmowania decyzji o charakterze heurystycznym
  - Jeżeli jeden z podcelów ze zbioru celów *jest spełniony w stanie początkowym*, to próbę jego osiągnięcia należy odłożyć na później - być może osiągnięcie jego nie będzie wymagało żadnych dodatkowych nakładów, gdyż spełnianie innych podcelów nie naruszy jego prawdziwości
  - Jeżeli jakiś podcel jest *łatwo zniweczyć* lub *łatwo osiągnąć*, to jego spełnienie należy odłożyć na później
- ©AM





## Regresja celów: *podsumowanie*

127

- Generowane drzewo poszukiwań planu jest bardzo rozgałęzione – algorytm wymaga analizy wszystkich możliwych uporządkowań podcelów (ewentualne nawroty!), jak również wszystkich możliwych sposobów ich spełnienia
- Analiza taka jest bezcelowa w problemach całkowicie dekomponowalnych – brak zależności między podcelami lub zależności te są bardzo słabe, więc kolejność w jakiej osiągamy wtedy podcele nie ma znaczenia
- Brak uniwersalnego mechanizmu, który wskazywałby najlepsze kierunki poszukiwania planu, co może prowadzić do analizy zupełnie marginalnych elementów planu, których występowanie w ostatecznym planie jest nieistotne

©AM

## Planowanie liniowe: *podsumowanie*

128

- W metodach planowania liniowego plan generowany jest sekwencyjnie dla kolejnych podcelów, tak długo aż nie zostanie osiągnięty stan docelowy
- plan prowadzący do spełnienia pewnego podcelu nie jest generowany dopóty, dopóki nie zakończy się planowanie dla podcelu bezpośrednio go poprzedzającego
- kompletny plan składa się z ciągu operacji prowadzących do pierwszego podcelu, po których następuje ciąg operacji spełniających drugi podcel, itd.
- kompletny plan jest liniowo uporządkowaną sekwencją kompletnych podplanów

©AM

## Wprowadzenie do Sztucznej Inteligencji

129

Wykład 7

Informatyka Studia Inżynierskie

©AM

## Planowanie nieliniowe: *charakterystyka*

130

- W metodach planowania nieliniowego generowanie operacji na rzecz jednego celu może przeplatać się z operacjami prowadzącymi do spełnienia innego podcelu
- Planowanie prowadzące do spełnienia pewnego podcelu może zostać przerwane w celu zbudowania planu dla innego podcelu, po czym możliwy jest powrót do poprzedniego celu planowania lub może rozpocząć się generacja planu dla jeszcze innego podcelu
- Nieliniowe konstruowanie planu przebiega niezależnie od liniowej (i późniejszej) realizacja planu
- Plan nie jest wykonywany zanim nie zostanie kompletnie zbudowany

©AM

## Planowanie nieliniowe: charakterystyka

131

- planowanie odbywa się w *przestrzeni planów* a nie w przestrzeni stanów (sytuacji)
  - rozpoczynamy od niepełnego, ogólnego planu
  - operatory służą do uszczegóławiania bądź modyfikacji planu takich jak: dodanie nowej operacji, zmiana kolejności operacji, wiązanie zmiennej wolnej itp.
  - rozbudowa planu kończy się, gdy otrzymamy kompletny plan prowadzący od  $S_0$  do  $S_g$  i nie jest istotne jaką ścieżką osiągnęliśmy rozwiązanie
- planowanie odbywa się *wstecz* (wybieramy operatory na podstawie osiągniętych przez nie celów), aby ograniczyć rozmiar przeszukiwanej przestrzeni planów
- ustalanie kolejności operatorów i wiązań zmiennych odbywa się zgodnie ze *strategią least-commitment*
- rezultatem końcowym planowania jest *częściowo uporządkowany plan* w przeciwieństwie do systemów liniowych, w których plan jest całkowicie uporządkowanym ciągiem operacji

©AM

## Strategia least-commitment

146

Strategia least-commitment planowania w przestrzeni planów oznacza, że:

- nie jest wymagane z góry podjęcie wszystkich decyzji dotyczących planu (kolejność operacji, wiązania zmiennych itp.)
- dokonywane wybory dotyczą tylko tych aspektów planu, które są istotne w danym momencie i których nie można uniknąć
- odkładanie decyzji tak długo jak to możliwe pozwoli zredukować liczbę nawrotów, których wystąpienie jest właśnie bezpośrednią konsekwencją błędnie podjętych decyzji (za wcześniej i przy zbyt ograniczonej wiedzy)

©AM

## Plan w przestrzeni przeszukiwania planów

147

- Jest częściowo uporządkowanym grafem
- Osiągany jest ścieżką, której nie musimy pamiętać w przeciwieństwie do planowania w przestrzeni stanów
- Jest reprezentacją zbioru kompletnych i w pełni uporządkowanych planów
- Jego wykonanie wymaga związania wszystkich zmiennych (konkretyzacja)
- Jego realizacja zwana linearyzacją planu jest jedną z wielu możliwych konkretyzacji planu częściowego

©AM

## Planowanie częściowo uporządkowane: *reprezentacja planu*

148

Plan składa się z:

- zbioru kroków; każdy krok reprezentuje operację, będącą fragmentem realizacji planu
- zbioru ograniczeń kolejnościowych reprezentowanych przez pary wyrażeń postaci  $S_i \prec S_j$ , które oznacza, że stan  $S_i$  musi wystąpić przed stanem  $S_j$  (niekoniecznie bezpośrednio)
- zbioru wiązań wartości zmiennych w postaci wyrażeń  $X=Y$ , gdzie zmienna  $X$  przyjmuje wartość  $Y$  (stałej lub zmiennej)
- zbioru zależności przyczynowych (ang. casual links) pod postacią wyrażeń  $S_i \xrightarrow{c} S_j$ , które oznacza, iż krok  $S_i$  „spełnia warunek”  $c$  kroku  $S_j$

©AM

## Planowanie częściowo uporządkowane: *operatory*

Plan zawsze zaczyna się od predefiniowanego operatora *Start* i kończy predefiniowanym *Finish*.

**Start:** pierwszy operator w planie (zawsze!); brak warunków stosowalności; efektem działania (lista ADD) jest dodanie predykatów, które są spełnione w stanie początkowym

**Finish:** ostatni operator w planie (zawsze!); brak efektów wykonania; warunkiem stosowalności (lista PRE...) jest zbiór predykatów, opisujących stan docelowy

**Start:** PRECONDITION: null  
DELETE: null  
ADD:  $S_0$

**Finish:** PRECONDITION:  $S_g$   
DELETE: null  
ADD: null

## Planowanie częściowo uporządkowane: *przykład*

Przykład planowania częściowo uporządkowanego dla zadania *ubierania skarpetek i butów*

### Operator:

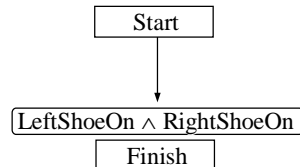
<b>RightSock:</b>	PRECONDITION: null	<b>LeftSock:</b>	PRECONDITION: null
DELETE: null		DELETE: null	
ADD: RightSockOn		ADD: LeftSockOn	
<b>RightShoe:</b>	PRECONDITION: RightSockOn	<b>LeftShoe:</b>	PRECONDITION: LeftSockOn
DELETE: null		DELETE: null	
ADD: RightShoeOn		ADD: LeftShoeOn	

### Operatory początkowe/predefiniowane:

<b>Start:</b>	PRECONDITION: null	<b>Finish:</b>	PRECONDITION: RightShoeOn $\wedge$ LeftShoeOn
DELETE: null		DELETE: null	
ADD: null		ADD: null	

## Planowanie częściowo uporządkowane: *przykład*

### Graf początkowy poszukiwania planu:

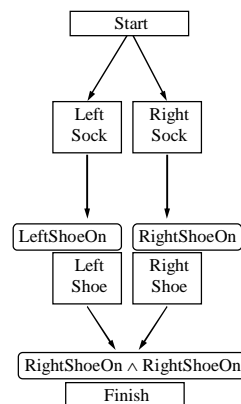


PLAN(STEPS: {  $S_1: Start()$   
 $S_2: Finish(PRECOND: LeftShoeOn \wedge RightShoeOn)$   
 ORDERINGS: {  $S_1 \prec S_2$  }  
 BINDINGS: { }  
 LINKS: { } )

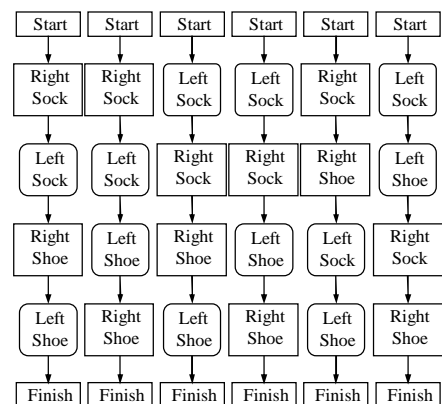
← Reprezentacja planu

## Planowanie częściowo uporządkowane: *przykład*

### Plan częściowo uporządkowany:

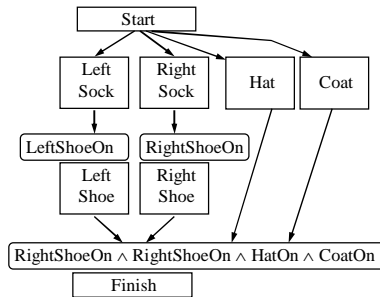


### Wszystkie plany całkowicie uporządkowane:



## Planowanie częściowo uporządkowane: *przykład*

Plan częściowo uporządkowany:



- Reprezentacja rozwiązania w postaci planu częściowo uporządkowanego jest bardziej zwięzła
- Każdy nowy krok w częściowym planie powoduje wykładniczy wzrost liczby linearyzacji planu
- *Przykład:* dodanie dwóch nowych operacji (*Hat* i *Coat*) będzie skutkowało liczbą 180 linearyzacji

©AM

## Planowanie częściowo uporządkowane: *reprezentacja rozwiązania*

Wynik planowania częściowo uporządkowanego, to acykliczny graf skierowany, w którym porządek nie wszystkich operacji jest zdeterminowany i nie wszystkie zmienne mają określoną wartość, więc musi on być:

- planem kompletnym, w którym każdy warunek  $c$  kroku  $O_j$  jest osiągnięty przez zastosowanie pewnej operacji  $O_i$  takiej, że  $O_i \prec O_j$ , i  $c \in \text{ADD}(O_i)$  oraz  $\neg \exists O_k, O_k \prec O_j \wedge c \in \text{DELETE}(O_k)$ , gdzie  $O_i \prec O_k \prec O_j$  jest jedną z możliwych linearyzacji planu
- planem spójnym, w którym nie występują sprzeczności w porządku operacji lub wiązaniu zmiennych tzn. nie zachodzi nigdy  $O_i \prec O_j \wedge O_j \prec O_i$  lub  $X=a$  i  $X=b$  (dla dwóch różnych stałych  $a$  i  $b$ )

©AM

## Przykład planowania nieliniowego

- **Zadanie:**  
*Zaplanować zakupy mleka, gazety i wiertarki wraz z powrotem do domu.*
- **Założenia:**
  - Wyruszamy na zakupy z domu i wracamy z nimi do domu*
  - Możemy wykonać tylko dwie operacje: Go (przemieszczanie) oraz Buy (kupowanie)*
  - W operacji kupowania ignorujemy kwestię kosztów (pieniędzy)*

©AM

## Przykład planowania nieliniowego

Predykaty opisu stanu:

**at(X)** - wskazuje aktualne położenie jako X (np. *home*, *cornershop* itp.)

**sells(X,Y)** - oznacza, że towar Y można zakupić w X

**have(X)** - oznacza, że posiadamy X

Operatory:

<b>Go(Y):</b>	PRECONDITION: at(X)	<b>Buy(X):</b>	PRECONDITION: at(S) $\wedge$ sells(S,X)
	DELETE: at(X)		DELETE: null
	ADD: at(Y)		ADD: have(X)

Operatory predefiniowane:

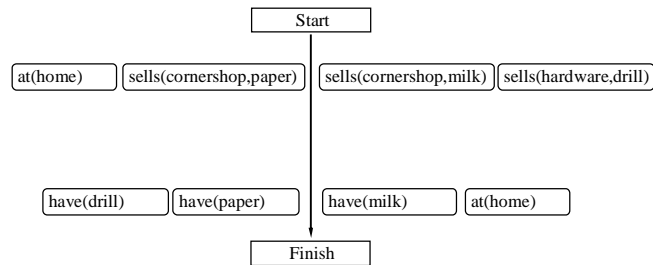
<b>Start:</b>	PRECONDITION: at(home) $\wedge$ sells(hardware,drill) $\wedge$ sells(cornershop,paper) $\wedge$ sells(cornershop,milk)	<b>Finish:</b>	PRECONDITION: have(drill) $\wedge$ have(milk) $\wedge$ have(paper) $\wedge$ at(home)
DELETE:	null	DELETE:	null
ADD:	null	ADD:	null

©AM

## Przykład planowania nieliniowego

158

Graf początkowy poszukiwania planu:

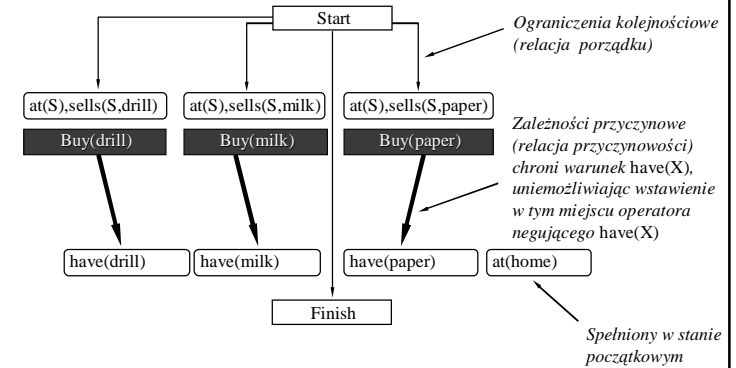


©AM

## Przykład planowania nieliniowego

159

Planowanie w przestrzeni planów:

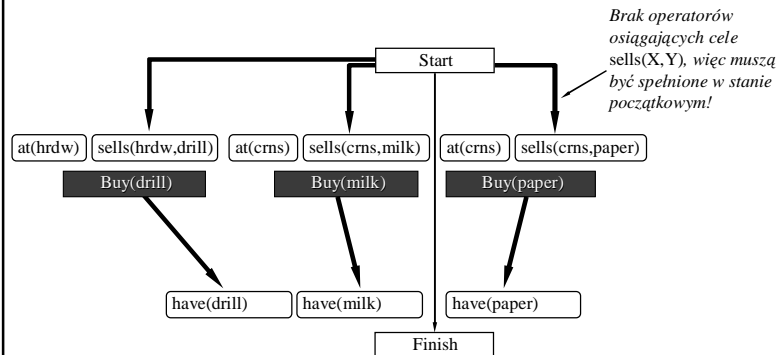


©AM

## Przykład planowania nieliniowego

160

Planowanie w przestrzeni planów: wybór celów have(drill), have(milk) oraz have(paper)



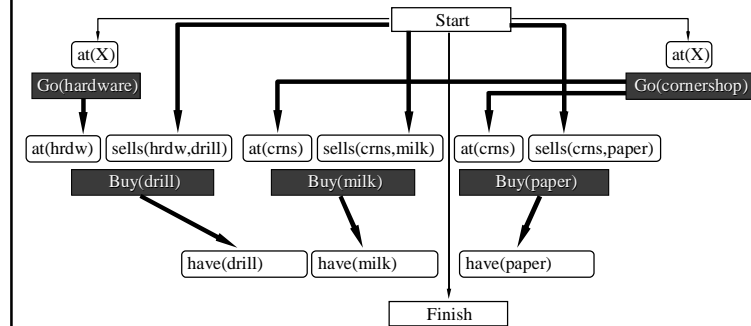
©AM

Skróty: crms=cornershop oraz hrdw=hardware

## Przykład planowania nieliniowego

161

Planowanie w przestrzeni planów: wybór celów at(hardware) oraz at(cornershop)

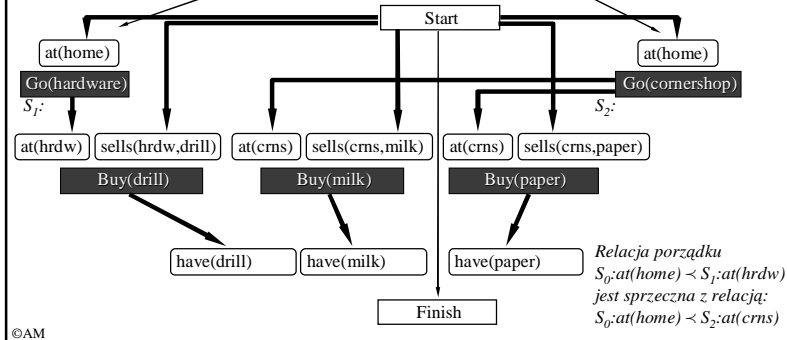


©AM

## Przykład planowania nieliniowego

Planowanie w przestrzeni planów: próba spełnienia celu  $at(X)$  przez  $X=home$

Na podstawie stanu początkowego  $X=home$  !

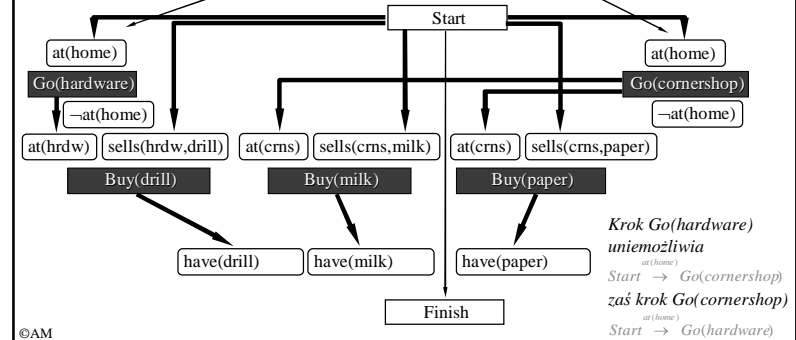


©AM

## Przykład planowania nieliniowego

Planowanie w przestrzeni planów: próba spełnienia celu  $at(X)$  przez  $X=home$

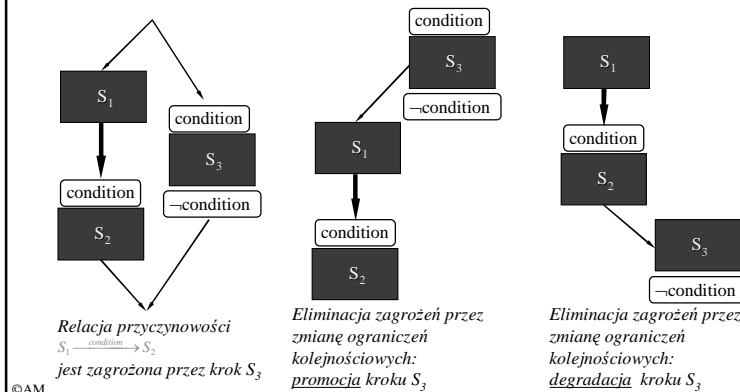
Na podstawie stanu początkowego  $X=home$  !



©AM

## Planowanie nieliniowe: obsługa tzw. zagrożeń

Planowanie w przestrzeni planów: ochrona zagrożonych relacji przyczynowości

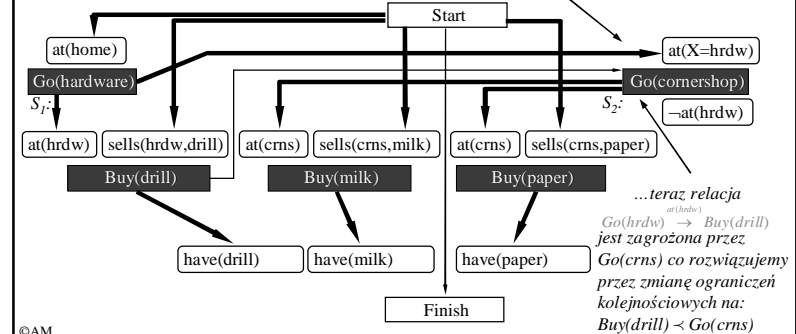


©AM

## Przykład planowania nieliniowego

Planowanie w przestrzeni planów: próba spełnienia celu  $at(X)$  dla  $Go(corneshop)$  przez  $X=hrdw$

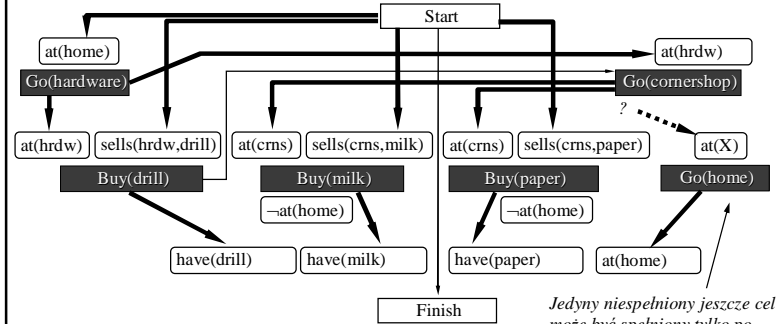
Ani degradacja ani promocja nie eliminują wzajemnego zagrożenia  $S_1$  i  $S_2$ . Tylko próba osiągnięcia w inny sposób jednego z warunków  $at(X)$  może coś zmienić ale ...



©AM

## Przykład planowania nieliniowego

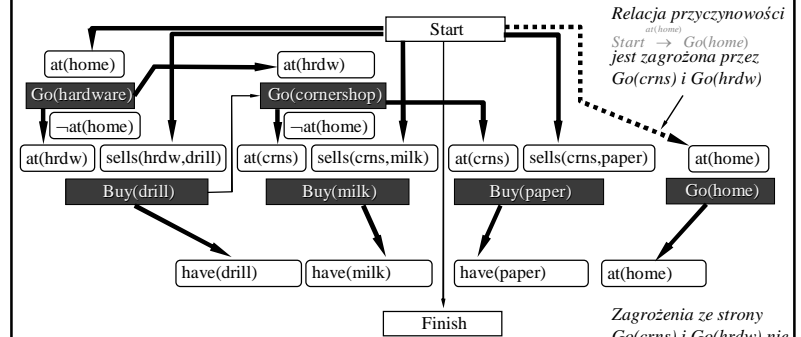
Planowanie w przestrzeni planów: Go(hrdw) i Go(crns) oznaczają, że mamy  $\neg\text{at}(\text{home})$  dla Finish



©AM

## Przykład planowania nieliniowego

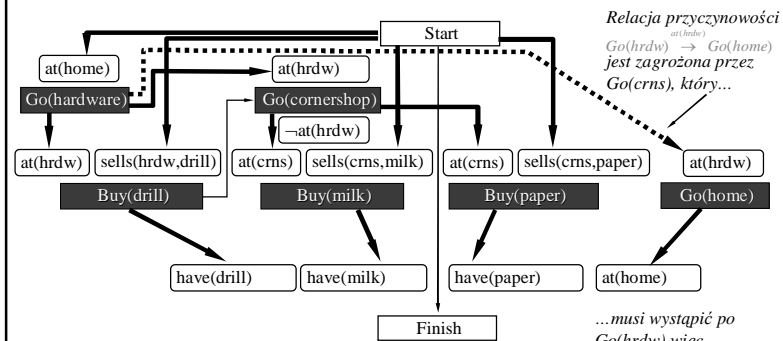
Planowanie w przestrzeni planów: próba spełnienia  $\text{at}(X)$  dla Go(home) przez krok Start ( $X=\text{home}$ )



©AM

## Przykład planowania nieliniowego

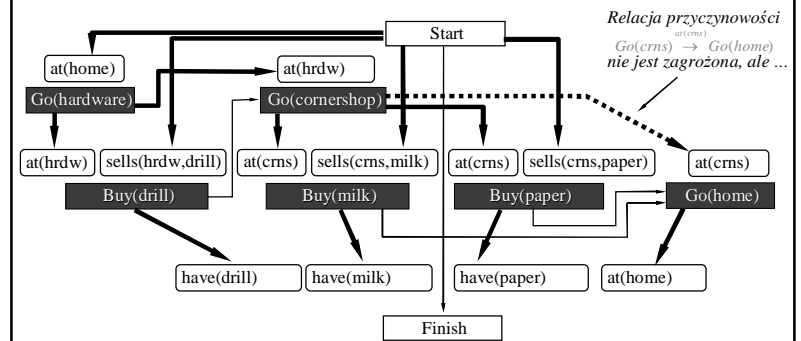
Planowanie w przestrzeni planów: druga próba spełnienia  $\text{at}(X)$  dla Go(home) przez  $X=\text{hardware}$



©AM

## Przykład planowania nieliniowego

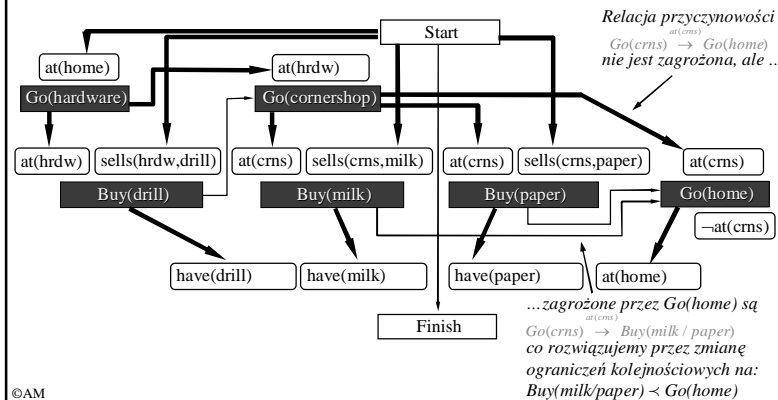
Planowanie w przestrzeni planów: trzecia próba spełnienia  $\text{at}(X)$  dla Go(home) przez  $X=\text{cornershop}$



©AM

## Przykład planowania nieliniowego

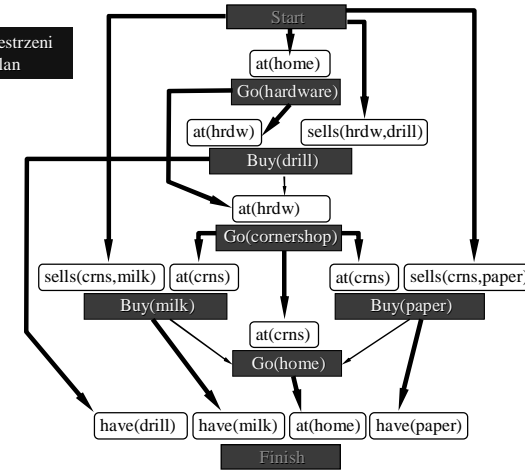
Planowanie w przestrzeni planów: trzecia próba spełnienia  $at(X)$  dla  $Go(home)$  przez  $X=corner-shop$



©AM

## Przykład planowania nieliniowego

Planowanie w przestrzeni planów: gotowy plan



©AM

## Algorytm POP – Partial Order Planner

- Zaczynamy od szkieletu planu w postaci stanu początkowego i końcowego
- W każdym podejmowana jest próba spełnienia warunku *precond* pewnego kroku  $S_{need}$  zawartego w planie poprzez wybór nowego operatora albo znalezienie odpowiedniego kroku w częściowym planie
- Znalezionej operator generuje nową relację przyczynowości
- Brak odpowiedniego operatora powoduje nawrót
- Nowa relacja przyczynowości wymaga sprawdzenia zagrożeń – albo nowy krok zagraża zapamiętanym relacjom przyczynowości, albo nowa relacja przyczynowości jest zagrożona przez kroki już będące w planie
- Jeżeli nie można usunąć zagrożenia mamy nawrót
- Warunkiem stopu jest brak niespełnionych warunków *precond* w planie

©AM

## Algorytm POP – Partial Order Planner

```
function POP( $S_{init}, S_{goal}, operators$ ): plan
  plan  $\leftarrow$  MINIMAL-PLAN( $S_{init}, S_{goal}$ )
  loop do
    if IS-SOLUTION(plan) then return plan
     $S_{need}, c \leftarrow$  SELECT-SUBGOAL(plan)
    CHOOSE-OP(plan, operators,  $S_{need}, c$ )
    RESOLVE_THREATS(plan)
  end
```

```
procedure RESOLVE-THREATS(plan)
  for each  $S_{threat}$  such that  $c \in DEL(S_{threat})$ 
    and a link  $S_i \xrightarrow{c} S_j$  in LINKS(plan) do
    choose either
      Promotion: add  $S_{threat} \prec S_i$  to ORDERINGS(plan)
      Demotion: add  $S_i \prec S_{threat}$  to ORDERINGS(plan)
    if not CONSISTENT(plan) then fail
  end
```

```
procedure CHOOSE-OP(plan, operators,  $S_{need}, c$ )
  choose a step  $S_{add}$  from operators or STEPS(plan)
  such that  $c \in ADD(S_{add})$ 
  if there is no such step then fail
  add casual link  $S_{add} \xrightarrow{c} S_{need}$  to LINKS(plan)
  add the ordering constraint  $S_{add} \prec S_{need}$ 
  to ORDERINGS(plan)
  if  $S_{add}$  is newly added step from operators then
    add  $S_{add}$  to STEPS(plan)
    add  $S_{init} \prec S_{add} \prec S_{goal}$  to ORDERINGS(plan)
  end
```

```
function SELECT-SUBGOAL(plan):  $S_{need}, c$ 
  find a plan step  $S_{need}$  in STEPS(plan)
  with precondition  $c$  unsatisfied
  return  $S_{need}, c$ 
end
```

©AM



## Algorytm POP – Partial Order Planner

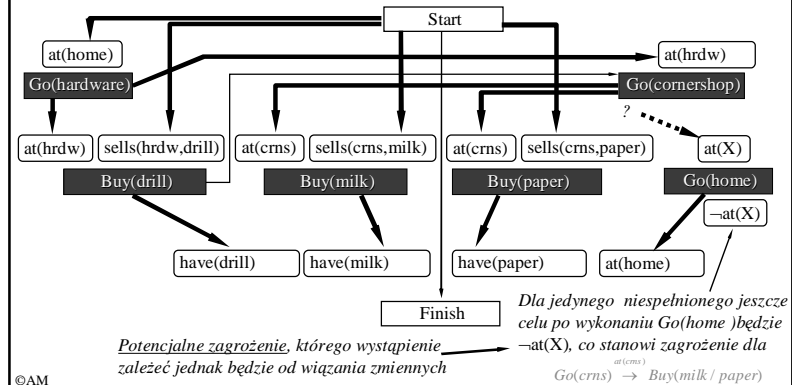
Istotne cechy algorytmu POP

- Nawroty w procesie konstrukcji planu dotyczą tylko mechanizmu wyboru operatora oraz mechanizmu eliminacji zagrożeń (zmiany ograniczeń kolejnościowych)
- Decyzje o wyborze warunków  $c$  do spełnienia nie są źródłem nawrotów – każdy warunek musi być w końcu spełniony, więc kolejność ich osiągania może mieć wpływ tylko na wydajność algorytmu (ale nie na końcowy efekt!)
- Planowanie jest regresywne
- Procedura planowania jest poprawna i pełna – każdy wynik jest rozwiązaniem zadania planowania, dla każdego zadania planowania zostanie znalezione rozwiązanie, o ile istnieje
- Algorytm jest niedeterministyczny – operacje *choose* oraz *fail* odpowiadają za decyzję sterującą i nawrót

©AM

## Przykład tzw. potencjalnego zagrożenia w trakcie planowanie nieliniowego POP

Planowanie w przestrzeni planów: Go(hrdw) i Go(crns) oznaczają, że mamy  $\neg at(home)$  dla Finish



©AM

## Planowanie z częściowo ukonkretnionymi operatorami

- W trakcie rozstrzygania zagrożeń (RESOLVE-THREATS) wpływ wybranego operatora na relacje przyczynowości w planie może zależeć od wiązania zmiennych na jego listach PRECOND, ADD i DEL
- Zagrożenie takie są określane mianem zagrożeń potencjalnych
- Zagrożenia potencjalne można eliminować jak najszybciej wymuszając przyjęcie wiązania zmiennej, które likwiduje to zagrożenie (np. wiązanie  $X=hrdw$  w  $\neg at(X)$  nie stanowi zagrożenia dla  $at(home)$ )
- Poprzez rozszerzenie języka opisu wiązań zmiennych o negację (np.  $X \neq home$ ) można wykluczyć pewne wartości w celu eliminacji zagrożenia bez nadawania konkretnej wartości zmiennej – mniej rygorystyczny mechanizm ograniczania  $X$
- Najłagodniejsze podejście opóźnia obsługę zagrożeń do momentu, gdy faktycznie one wystąpią, ale wymaga ono najbardziej złożonej procedury weryfikacji poprawności planu

©AM

## Opóźniona obsługa zagrożeń potencjalnych w algorytmie POP

Odkładanie obsługi potencjalnych zagrożeń do momentu ich faktycznego wystąpienia wymaga w każdym wywołaniu procedury rozstrzygania sprawdzania zawsze wszystkich relacji przyczynowych i wszystkich kroków, które mogą im zagrozić po uwzględnieniu wiązań (SUBST)

```

procedure RESOLVE-THREATS(plan)
for each link  $S_i \leftarrow c \rightarrow S_j$  in LINKS(plan) do
  for each  $S_{threat}$  in STEPS(plan) do
    for each  $c'$  such that  $c' \in DEL(S_{threat})$  do
      if SUBST(BINDINGS(plan),  $c$ ) =
        =SUBST(BINDINGS(plan),  $c'$ ) then
        choose either
        Promotion: add  $S_{threat} \leftarrow S_i$  to ORDERINGS(plan)
        Demotion: add  $S_j \leftarrow S_{threat}$  to ORDERINGS(plan)
      if not CONSISTENT(plan) then fail
end
  
```

©AM

```

procedure CHOOSE-OP(plan, operators,  $S_{need}, c$ )
  choose a step  $S_{add}$  from operators or STEPS(plan)
  that has  $c_{add} \in ADD(S_{add})$ 
  and such that  $u = UNIFY(c, c_{add}, BINDINGS(plan))$ 
  if there is no such step then fail
  add  $u$  to BINDINGS(plan)
  add casual link  $S_{add} \leftarrow c \rightarrow S_{need}$  to LINKS(plan)
  add  $S_{add} \sim S_{need}$  to ORDERINGS(plan)
  if  $S_{add}$  is newly added step from operators then
    add  $S_{add}$  to STEPS(plan)
    add  $S_{init} \leftarrow S_{add} \leftarrow S_{goal}$  to ORDERINGS(plan)
  end
  
```

Obsługa wiązań zmiennych opiera się na algorytmie uzgadniania (UNIFY) i wymaga ich zapamiętywania w planie pod polem BINDINGS

## Definicja rozwiązania problemu planowania dla częściowo ukonkretnionych operatorów

178

- Pojawienie się częściowo związanych operatorów w planie wymaga uwzględnienia wpływu wiązań na możliwość osiągnięcia rozwiązania jakim jest częściowo uporządkowany plan, w którym:

każdy warunek  $c$  kroku  $O_j$  jest osiągany przez zastosowanie pewnej operacji  $O_i$

- takiej, że  $O_i \prec O_j$  i  $\exists c' \in \text{ADD}(O_i)$ , który koniecznie unifikuje się z  $c$  oraz
  - $\neg \exists O_k$  takie, że  $O_i \prec O_k \prec O_j$  w jednej z możliwych linearyzacji planu i  $c \in \text{DELETE}(O_k)$ ,
- W algorytmie POP za warunek 1 odpowiada procedura CHOOSE-OPERATOR zaś za warunek 2 – procedura RESOLVE-THREATS

©AM

## Planowanie nieliniowe: *podsumowanie*

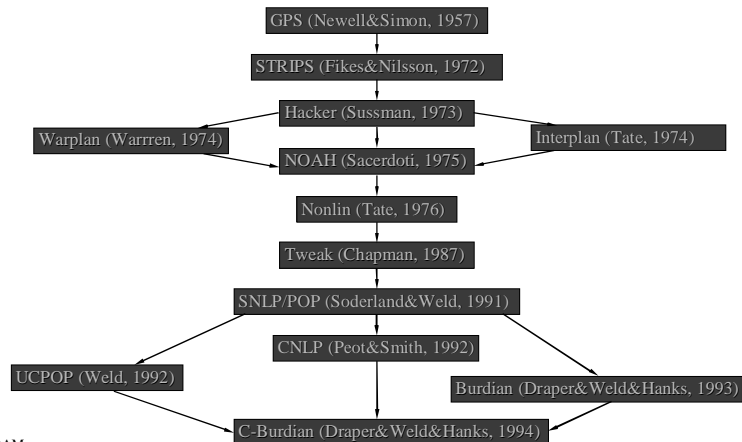
179

- System NONLIN – casual links, mechanizmy nawrotu, wykrywanie zagrożeń/konfliktów (ang. threats), ich eliminacja, nawroty typu *dependency-directed*, operatory modyfikacji planu
- System TWEAK – formalizacja planowania z wykorzystaniem mechanizmów spełniania ograniczeń (ang. constraint posting), gwarantująca poprawność i pełność algorytmu
- System SNLP/POP – systematyczne częściowe planowanie, większa ekspresywność języka opisu celów

©AM

## Systemy planowania: *zarys rozwoju*

180



©AM