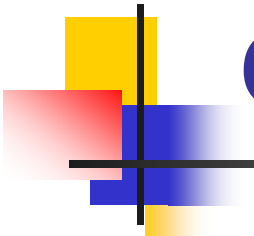
Inteligenta Artificiala

Universitatea Politehnica Bucuresti
Anul universitar 2013-2014

Adina Magda Florea



Curs nr. 8

Planificare automata

- PA – caracteristici
- Planificare liniara in sistemul STRIPS
- Grafuri de planificare
- Planificare neliniara – TWEAK
- Planificare ierarhica
- Planificare contingenta

1. PA - caracteristici

Rationament de bun simt:

- Problema cadrului, problema calificarii si problema ramificarii

Descompunerea problemelor in subprobleme

Tipuri de planificare:

- liniara
- neliniara
- ierarhica
- contingenta

PA – caracteristici - cont

- **Reprezentare**
  - Reprezentarea cunostintelor in problemele de planificare
  - Reprezentarea starilor cautarii
  - Operatori de plan
  - Predicate
  - Axiome
- **Cautarea solutiei**
  - Forward planning/search
  - Backward planning / regression

PA – caracteristici - cont

- Actiuni (operatori de plan) cu preconsitii si postconditii
- Cautare inainte in spatiul starilor
- Cautare inapoi (regresie) in spatiul starilor

PA Forward

- Algoritm Forward(S,Scopuri,A, Cale)

1. **daca** S staisface scopurile din Scopuri
   atunci intoarce Cale
   altfel
      1.1 Act=alege din A o actiune cu precond satisfacute de S
      1.2 **daca** nu exista A **atunci intoarce** *Fail*
      1.3 **altfel** fie S'=efectul simularii executiei Act in S
                intoarce
                Forward(S',Scopuri,A,conc(Cale,A))

sfarsit

PA Backward

- Algoritm Regresie(S,Scopuri,A, Cale)

1. **daca** S staisface scopurile din Scopuri

   atunci intoarce Cale

   altfel

   1.1 Act=alege din A o actiune cu efectu; satisfacut in S

   1.2 G=regresia scopurilor prin A

   1.3 **daca** nu exista A **sau** G este nedefinit **sau** G include Scopuri **atunci intoarce** *Fail*

   1.4 **altfel intoarce**

   Regresie(S,Scopuri,A,conc(A,Cale))

sfarsit

2. Planificare liniara in sistemul STRIPS

- Operatori de plan

  - *Actiune* care reprezinta actiunea asociata operatorului.

  - *Lista Preconditiilor* ce contine formulele care trebuie sa fie adevarate intr-o stare a problemei pentru ca operatorul sa poata fi aplicat - LP.

  - *Lista Adaugarilor* ce contine formulele care vor deveni adevarate dupa aplicarea operatorului - LA.

  - *Lista Eliminarilor* ce contine formulele care vor deveni false dupa aplicarea operatorului - LE.

2.1 Reprezentarea STRIPS

- Operatori de plan

  STACK(x,y), UNSTACK(x,y), PICKUP(x), PUTDOWN(x)

- Predicate:

  ON(x,y), ONTABLE(x), CLEAR(x), HOLD(x), ARMEMTY

- Axiome:

$$(\exists x)\ (\mathrm{HOLD}(x)) \rightarrow \sim \mathrm{ARMEMPTY}$$

$$(\forall x)\ (\mathrm{ONTABLE}(x) \rightarrow \sim (\exists y)\ (\mathrm{ON}(x,y)))$$

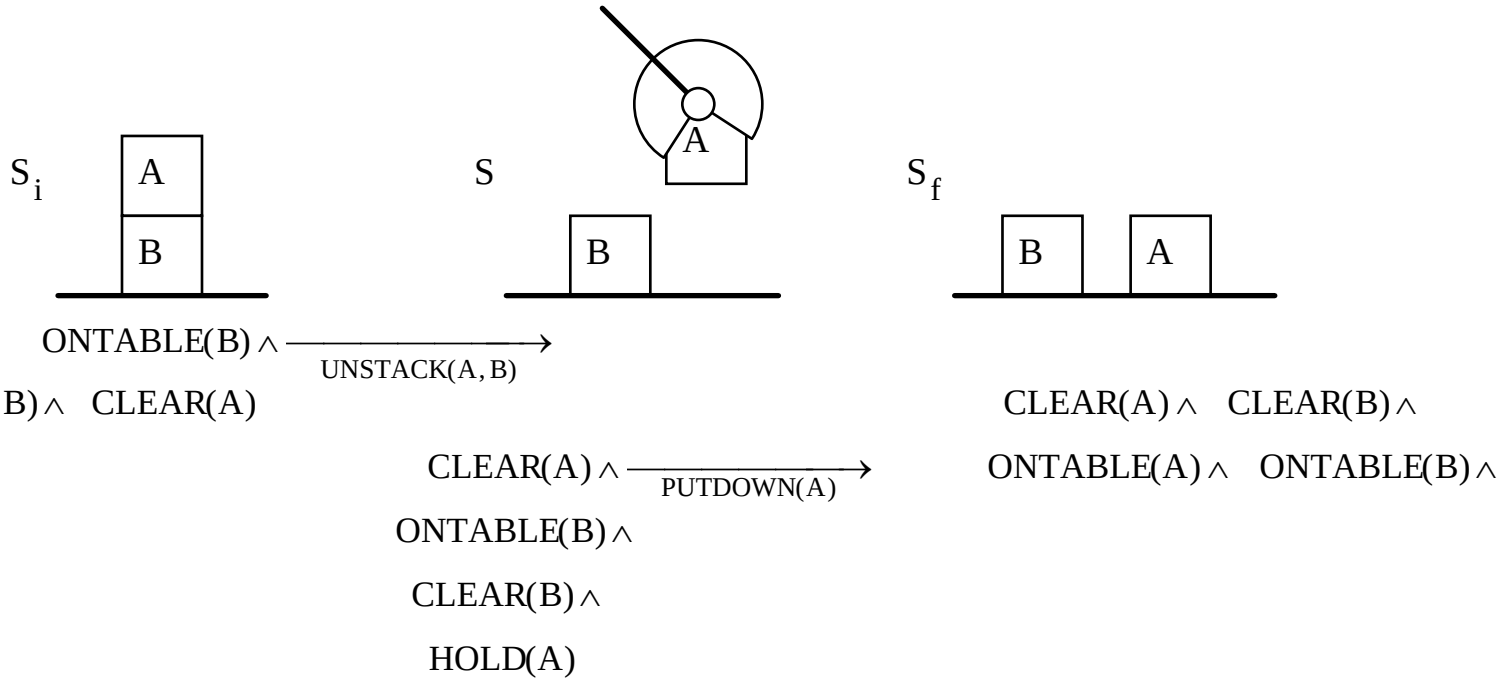
$$(\forall x)\ (\sim (\exists y)\ (\mathrm{ON}(y,x)) \rightarrow \mathrm{CLEAR}(x))$$

Reprezentarea STRIPS - cont

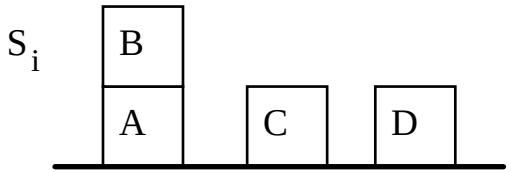
- Operatori de plan

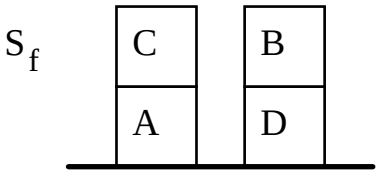
STACK(x, y)	LP:	$CLEAR(y) \wedge HOLD(x)$
	LE:	CLEAR(y)HOLD(x)
	LA:	ON(x,y)ARMEMPTY
UNSTACK (x, y)	LP:	$ON(x, y) \wedge CLEAR(x) \wedge ARMEMPTY$
	LE:	$ON(x, y) \wedge ARMEMPTY$
	LA:	$HOLD(x) \wedge CLEAR(y)$
PICKUP(x)	LP:	$CLEAR(x) \wedge ONTABLE(x) \wedge ARMEMPTY$
	LE:	$ONTABLE(x) \wedge ARMEMPTY$
	LA:	HOLD(x)
PUTDOWN (x)	LP:	HOLD(x)
	LE:	HOLD(x)
	LA:	$ONTABLE(x) \wedge ARMEMPTY$

2.2 Executia planului



2.3 Functionare STRIPS





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

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

Stiva 1

$ON(C,A)$

$ON(B,D)$

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

Stiva 2

$ON(B,D)$

$ON(C,A)$

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

Functionare STRIPS - cont

/* pentru realizarea scopului ON(C,A) */

STACK(C, A)

$ON(B,D)$

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

/* preconditiile operatorului STAC(C,A)*/

$CLEAR(A)$

$HOLD(C)$

$CLEAR(A) \wedge HOLD(C)$

STACK(C, A)

$ON(B,D)$

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

$S_1 \rightarrow S_2$: $ONTABLE(A) \wedge ONTABLE(C) \wedge ONTABLE(D) \wedge ON(B,D) \wedge ARMEMPTY$

$Plan = (UNSTACK(B,A), STACK(B,D))$

S_4: $ONTABLE(A) \wedge ONTABLE(D) \wedge ON(B,D) \wedge ON(C,A) \wedge ARMEMPTY$

$Plan = (UNSTACK(B,A), STACK(B,D), PICKUP(C), STACK(C,A))$

2.4 Algoritm STRIPS

- Variabila **S** - memoreaza descrierea starii curente a universului problemei;
- **Stiva** - memoreaza stiva de scopuri satisfacute pe calea curenta de cautare;
- **Scopuri** - pastreaza lista scopurilor nesatisfacute pe calea curenta;
- Structura **Operator** avand campurile: *Actiune*, *Preconditii*, *ListaAdaugari* si *ListaEliminari*

(Operator.Preconditii)

2.4 Algoritm STRIPS

Algoritm: Planificare liniara in STRIPS

SatisfacereScopuri (Scopuri, S, Stiva)

1. **pentru** fiecare Scop din Scopuri **executa**
   1.1. StareNoua ← RealizezaScop(Scop, S, Stiva)
   1.2. **daca** StareNoua = INSUCCES
   atunci intoarce INSUCCES
2. **daca** toate scopurile din Scopuri sunt satisfacute in starea StareNoua
   atunci intoarce StareNoua
3. **altfel intoarce** INSUCCES

sfarsit.

Algoritm STRIPS - cont

RealizeazaScop (Scop, S, Stiva)

1. **daca** Scop este marcat satisfacut in starea S
   atunci intoarce S
2. **daca** Scop apartine Stiva
   atunci intoarce INSUCCES
3. OperatoriValizi ←{O | O poate satisface scopul Scop}
4. **pentru** fiecare operator O din OperatoriValizi **executa**
   4.1. StareNoua ← AplicaOperator(O, S, Stiva ∪{Scop})
   4.2. **daca** StareNoua <> INSUCCES
   atunci
   4.2.1. Marcheaza scopul Scop satisfacut in starea StareNoua
   4.2.2. **intoarce** StareNoua
5. **intoarce** INSUCCES

sfarsit.

Algoritm STRIPS - cont

AplicaOperator (Operator, Stare, Stiva)

1. StareNoua←SatisfacereScopuri(Operator.Preconditii, Stare, Stiva)
2. **daca** StareNoua <> INSUCCES

   atunci

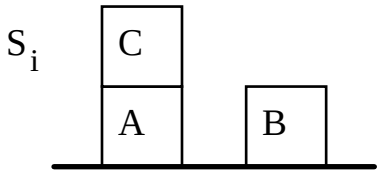
   2.1. adauga Operator.Actiune la Plan

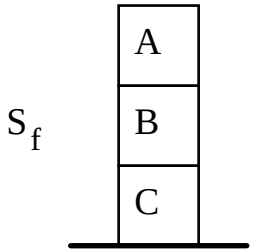
   2.2. StareNoua ← StareNoua – Operator.ListaEliminari

   2.3. **intoarce** StareNoua ∪ Operator.ListaAdaugari
3. **altfel intoarce** INSUCCES

sfarsit.

2.5 Anomalia lui Sussman





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

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

Stiva 1

Stiva 2

$ON(A,B)$

$ON(B,C)$

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

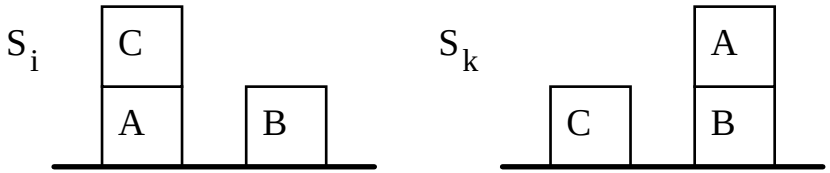
$ON(B,C)$

$ON(A,B)$

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

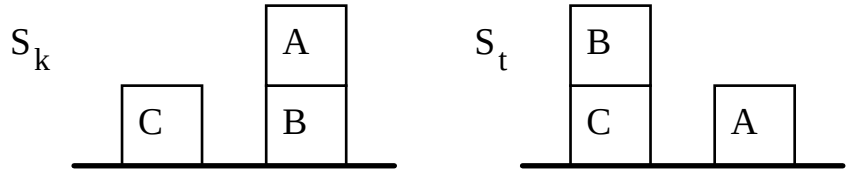
Anomalia lui Sussman - cont

Stiva 1



$$S_i \rightarrow S_k: \ ONTABLE(B) \wedge ON(A,B) \wedge ONTABLE(C) \wedge ARMEMPTY$$

$$Plan_{S_i \rightarrow S_k} = (UNSTACK(C,A), PUTDOWN(C), PICKUP(A), STACK(A,B))$$



$$S_k \rightarrow S_t: \ ON(B,C) \wedge ONTABLE(A) \wedge ONTABLE(C) \wedge ARMEMPTY$$

$$Plan_{S_k \rightarrow S_t} = (UNSTACK(A,B), PUTDOWN(A), PICKUP(B), STACK(B,C))$$

$$S_t \rightarrow S_f: \ ON(A,B) \wedge ON(B,C)$$

$$Plan_{S_t \rightarrow S_f} = (PICKUP(A), STACK(A,B))$$

$$Plan = (UNSTACK(C,A), PUTDOWN(C), PICKUP(A), STACK(A,B), UNSTACK(A,B)$$

$$PUTDOWN(A), PICKUP(B), STACK(B,C), PICKUP(A), STACK(A,B))$$

3. Grafuri de planificare

- Se aplica operatorilor instantiati (logica propozitiilor)
- Un graf de planificare este un **graf orientat organizat pe niveluri**:
  - nivel S_0 (S_i) pt starea S_0 – noduri care reprezinta literali adevarati in S_0 (S_i)
  - nivel A_0 (A_i) – noduri care reprezinta actiuni ce pot fi aplicate in S_0 (S_i)
- S_i – toti literalii care pot fi adevarati la momentul ***i*** in fct de actiunile executate anterior
- A_i – toate actiunile care pot fi executate deoarece au preconditiile satisfacute

3.1 Restrictii in GP

- Se adauga si actiuni de persistenta – **no-op**
- Conflicte intre actiuni ce nu pot fi executate impreuna – excludere mutuala – **legaturi mutex**
- Conflicte intre literari **-** – **legaturi mutex**
- **Graf stabilizat**
- A_i – toate actiunile in S_i + restrictii
- S_i – toti literalii adevarati pt orice alegere posibila de actiuni pe nivelul A_{i-1} + restrictii

Restrictii in GP

- Legaturi (restrictii) **mutex intre actiuni**:
  - **postconditii inconsistente (PI)** – o actiune neaga postconditia alteia
  - **destructivitate/interferente (D)** – postconditia unei actiuni este negarea preconditiei alteia
  - **necesitati competitive (NC)** – preconditia unei actiuni este mutual exclusiva cu preconditia alteia
- Legaturi (restrictii) **mutex intre literali**:
  - un literal si acelasi literal negat
  - fiecare pereche de actiuni care pot adauga cei 2 literali sunt mutex

Exemplu

- Exemplu (Credit: S. Russel & P. Norvig: Artificial Intelligence: A Modern Approach, Prentice Hall, 2009)

Actiune **Eat(Cake)**

Preconditii: Have(Cake)

Postconditii: $\neg$Have(Cake) $\wedge$ Eaten(Cake)

Actiune **Bake(Cake)**

Preconditii: $\neg$ Have(Cake)

Postconditii: Have(Cake)

S_i

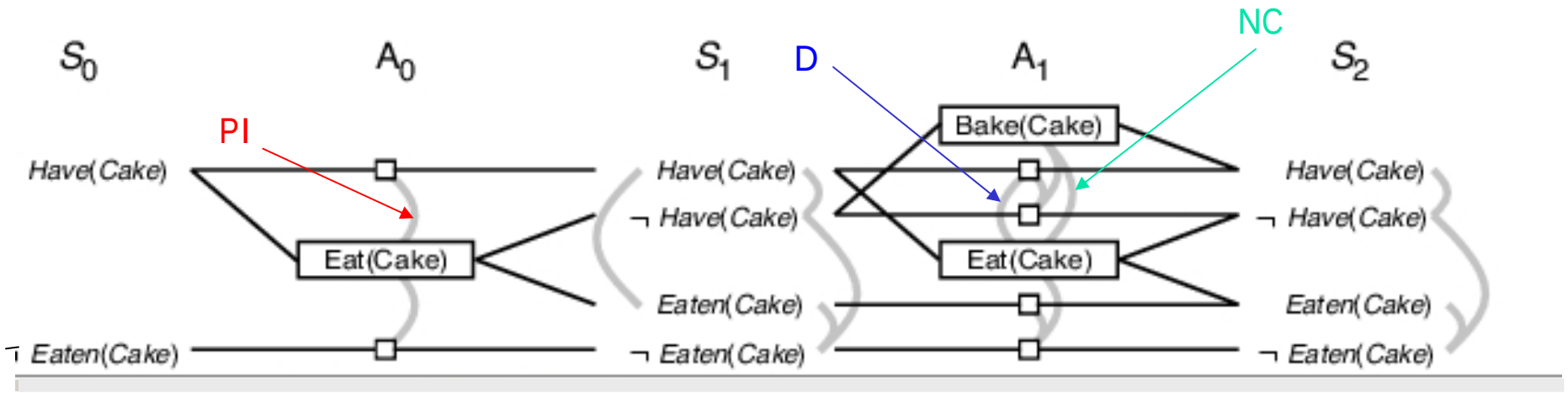
Have(Cake)

$\neg$Eaten(Cake)

S_f

Have(Cake) $\wedge$ Eaten(Cake)

Exemplu



- Figure Credit: S. Russel & P. Norvig: Artificial Intelligence: A Modern Approach, Prentice Hall, 2009

3.2 Despre GP

- ***l*** literali si ***a*** actiuni
- S_i are max ***l*** noduri si l^2 legaturi mutex
- A_i are max ***a+l*** noduri (inclusiv no-op) si $(a+l)^2$ legaturi mutex si ***2(al+l)*** legaturi preconditi si efect
- Un graf cu ***n*** niveluri are o dimensiune $O(n(a+l)^2)$

Despre GP

- Folosit si pt **estimari euristice**
- Costul necesar pt a satisface g_i din starea scop S_f
- **Costul nivel** al lui g_i
- Estimare cost **conjunctie de scopuri**
  - nivel-maxim – maxim nivel g_i
  - nivel-suma – suma niveluri g_i
  - nivel set – nivelul in care apar toate g_i a.i. intre nici o pereche sa nu exsite legaturi mutex

3.3 Algoritm

Algoritm: Planificare cu graf de planificare

GrafPlan(problema)

1. graf ← graf_initial (problema)
2. scopuri ← conjunctie(literali scop)
3. nogoods ← o tabela hash vida
4. **pentru** j ← 0 la infinit **executa**
    4.1 **daca** scopuri ne-mutex in S_j din graf **atunci**
        - solutie ← Extrage_Solutie(graf,scopuri,niv(graf),nogoods)
        - **daca** solutie ≠esec **atunci** intoarce solutie
    4.2 **daca** graf **si** nogoods s-au stabilizat **atunci intoarce** esec
    4.3 graf ← Expandeaza_Graf(graf, problema)

sfarsit

Algoritm

- **nogoods –** daca Extrage_Solutie nu gaseste solutie pt o multime de scopuri pe un nivel ***l*** atnci se memoreaza perechea (*l*, scopuri) ca un nogood

- **Extrage_Solutie – CSP**
  - Variabile: actiuni pe fiecare nivel
  - Valori variabile: *in* sau *out* (in plan)
  - Restrictii: legaturi mutex, preconditii scopuri

Algoritm

- **Extrage_Solutie – cautare**
  - Stare initiala – S_n impreuna cu scopuri problema
  - Actiuni intr-o stare pe nivel S_i – un subset de actiuni din A_{i-1} fara mutex si care au ca postconditii scopurile din S_i
  - Se trece in starea S_{i-1} care are ca subscopuri preconditiile actiunilor selectate
  - Terminare – sa se ajunga la o stare din S_0 a.i toate scopurile sunt satisfacute

- **Euristici**
  - alege literalul cu costul cel mai mare
  - pentru a satisface acel literal alege actiuni a.i. suma costurilor nivel a preconditiilor sa fie minima

Exemplu

(Credit: S. Russel & P. Norvig: Artificial Intelligence: A Modern Approach, Prentice Hall, 2009)

Actiune **Remove(obj,loc)**

Preconditii: At(obj,loc)

Postconditii: $\neg$ At(obj,loc) $\wedge$ At(obj,Ground)

Actiune **PutOn(t,Axle)**

Preconditii: Tire(t) $\wedge$ At(t,Ground) $\wedge \neg$ At(Flat,Axle)

Postconditii: $\neg$ At(t,Ground) $\wedge$ At(t,Axle)

Actiune **LeaveOvernight**

Preconditii:

Postconditii: $\neg$ At(Spare,Ground) $\wedge \neg$ At(Spare,Axle) $\wedge \neg$ At(Spare,Trunk)
$\wedge \neg$ At(Flat,Ground) $\wedge \neg$ At(Flat,Axle) $\wedge \neg$ At(Flat,Trunk)

S_i

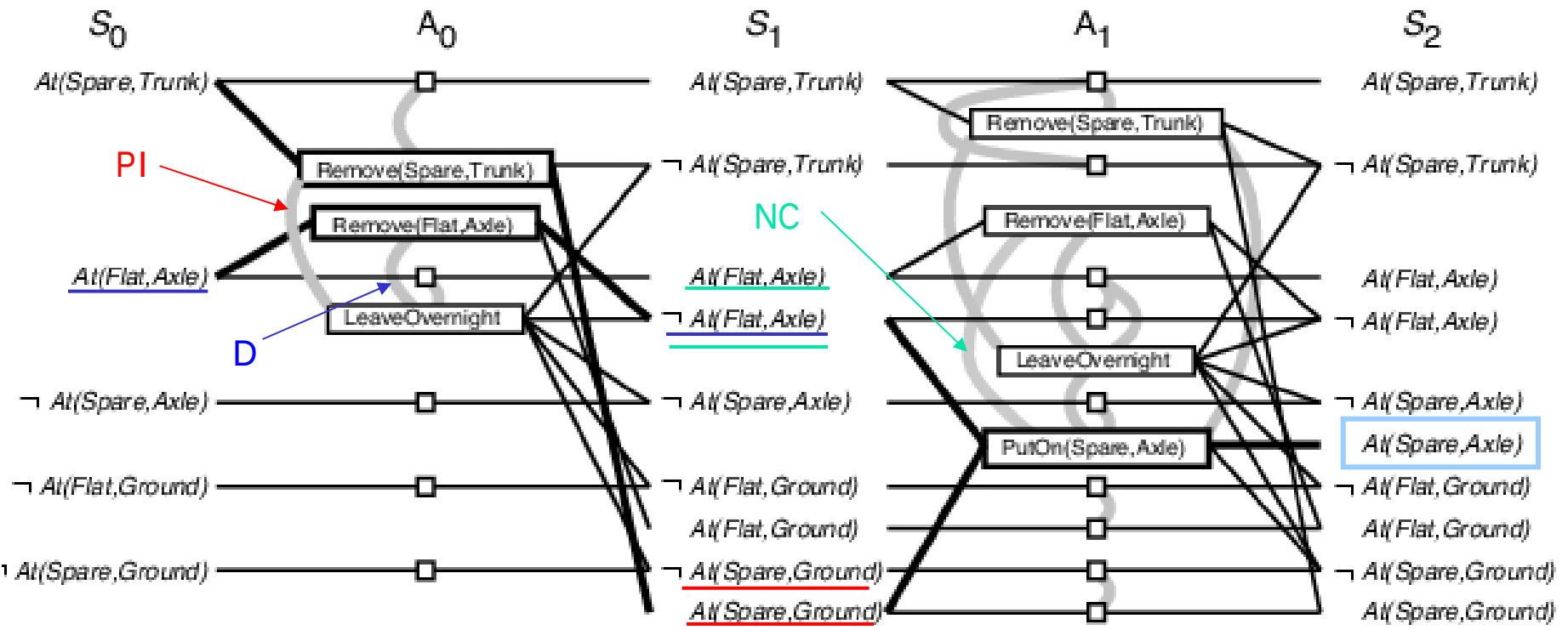
Tire(Flat), Tire(Spare)

At(Flat, Axle), At(Spare,Trunk)

S_f

At(Spare,Axle)

Exemplu



- Picture Credit: S. Russel & P. Norvig: Artificial Intelligence: A Modern Approach, Prentice Hall, 2009

4. Planificare neliniara - TWEAK

- Inregistrarea restrictiilor

  (temporale, unificare/codesemnare)
- Nivelul de reprezentare a planului;
- Nivelul de modificare a planului pentru a obtine un plan care realizeaza scopul problemei;

4.1 Reprezentarea TWEAK

Actiune: $\mathrm{STACK}(x, y)$

Preconditii: $\mathrm{CLEAR}(y) \wedge \mathrm{HOLD}(x)$

Postconditii: $\mathrm{ARMEMPTY} \wedge \mathrm{ON}(x, y) \wedge \sim \mathrm{CLEAR}(y) \wedge \sim \mathrm{HOLD}(x)$

Actiune: $\mathrm{PICKUP}(x)$

Preconditii: $\mathrm{CLEAR}(x) \wedge \mathrm{ONTABLE}(x) \wedge \mathrm{ARMEMPTY}$

Postconditii: $\mathrm{HOLD}(x) \wedge \sim \mathrm{ONTABLE}(x) \wedge \sim \mathrm{ARMEMPTY}$

4.2 Sinteza planului

Operatii de modificare a planului:

(1) *adaugarea de pasi* este operatia prin care se creaza noi pasi care se adauga la plan;

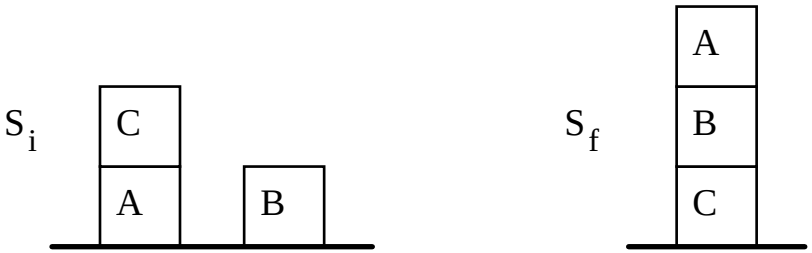
(2) *promovarea* este operatia de stabilire a unei ordonari (temporale) intre doi pasi de plan;

(3) *legarea simpla* este operatia de atribuire de valori variabilelor pentru a valida preconditiile unui pas de plan;

(4) *separarea* este operatia de impiedicare a atribuirii anumitor valori unei variabile;

(5) *eliminarea destructivitatii* este operatia de introducere a unui pas S3 (un pas deja existent in plan sau un pas nou) intre pasii S1 si S2, in scopul de a adauga un fapt invalidat de pasul S1 si necesar in pasul S2 (S1 amenintare pt S2).

Sinteza planului - cont



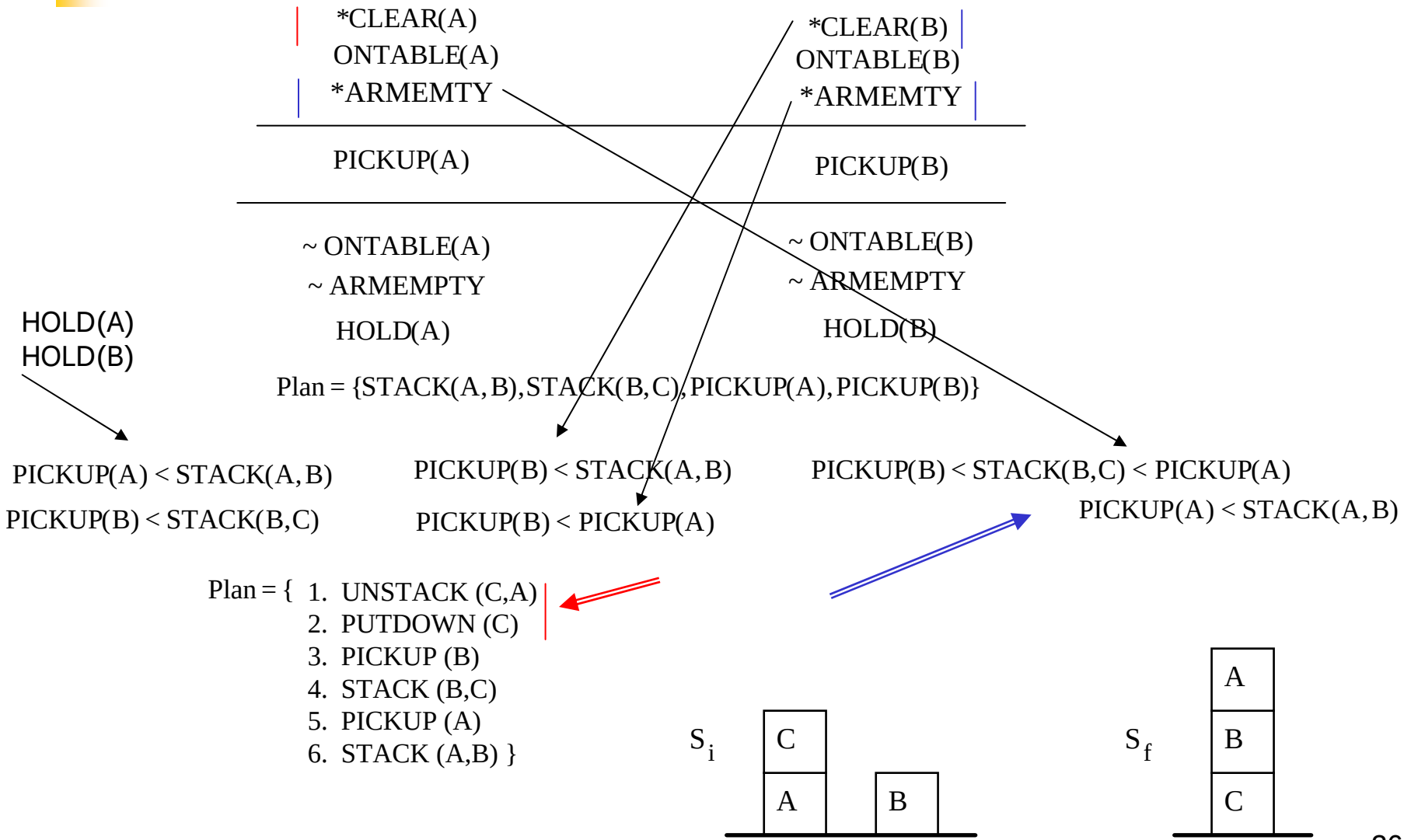
ON(C,A) ∧
ONTABLE(A) ∧
ONTABLE(B) ∧
ARMEMPTY

ON(A,B) ∧
ON(B,C)

CLEAR(B) *HOLD(A)	CLEAR(C) *HOLD(B)
STACK(A,B)	STACK(B,C)
ON(A,B) ~ CLEAR(B) ~ HOLD(A)	ON(B,C) ~ CLEAR(C) ~ HOLD(B) ARMEMTY

Plan = {STACK(A,B),STACK(B,C)}

Sinteza planului - cont



4.3 Algoritm TWEAK

Algoritm: Planificare neliniara in TWEAK

1. Initializeaza Plan ← {}
2. Initializeaza S cu multimea formulelor care definesc starea scop
3. **cat timp** S <> {} **executa**
   - 3.1. Alege si elimina o formula F din S
   - 3.2. **daca** F nu este satisfacuta in starea curenta

     atunci
     - 3.2.1. Alege o operatie de modificare a planului
     - 3.2.2. Aplica operatia si adauga efectul ei in Plan

Algoritm TWEAK - cont

3.3. Verifica pentru toti pasii din Plan satisfacerea preconditiilor

3.4. **pentru** fiecare preconditie nesatisfacuta a unui pas din Plan **executa**

Adauga preconditia la S

4. Genereaza ordinea totala a elementelor din Plan pe baza relatiilor de ordine individuale

5. **daca** planul Plan este partial

atunci

5.1. Instantiaza variabilele planului

5.2. Transforma arbitrar ordinea partiala in ordine totala

sfarsit.

4.4 Modelul formal al planificarii

- O formula este adevarata intr-o stare daca unifica cu o formula care face parte din stare respectiva.
- Un pas de plan *afirma* o formula in starea sa de iesire daca formula unfica cu o postconditie a pasului.
- Un pas de plan *infirma* o formula in starea sa de iesire daca afirma negarea acelei formule.
- Un pas de plan poate fi executat numai daca toate preconditiile sale sunt adevarate in starea sa de intrare.
- Starea de iesire a pasului de plan este starea de intrare din care se sterg formulele infirmate de catre pasul de plan si se adauga formulele afirmate de pasul de plan.
- Ordinea de efectuare a operatiilor de stergere si adaugare este importanta.

Modelul formal al planificarii - cont

Criteriul adevarului necesar

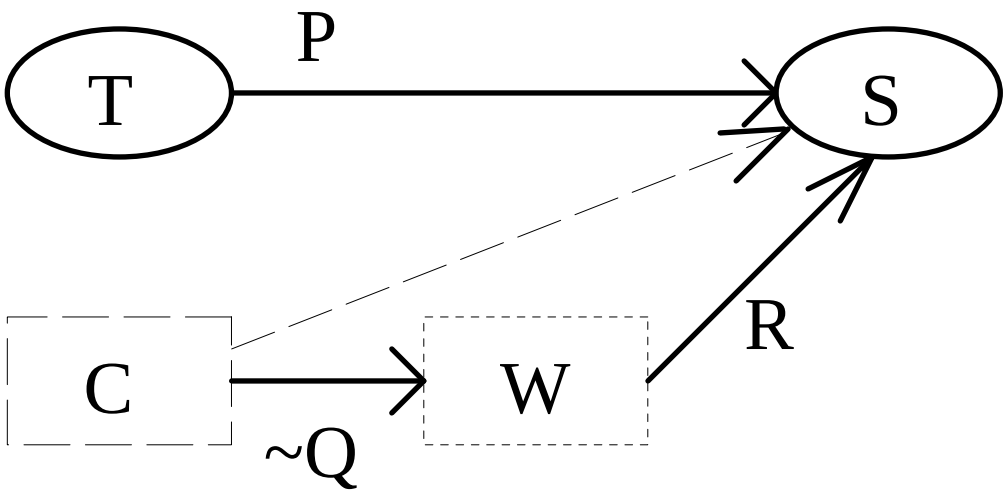
O formula **P** este necesar adevarata intr-o stare S daca si numai daca se indeplinesc urmatoarele doua conditii:

(1) exista o stare T egala cu/sau necesar anterioara lui S in care **P** este necesar afirmata (adaugata);

(2) pentru fiecare pas **C** posibil de executat inaintea lui S si pentru fiecare formula Q care poate unifica cu **P** pe care C o infirma, exista un pas W necesar intre **C** si S care afirma R, R fiind o formula pentru care R si **P** unifica ori de cite ori **P** si Q unifica.

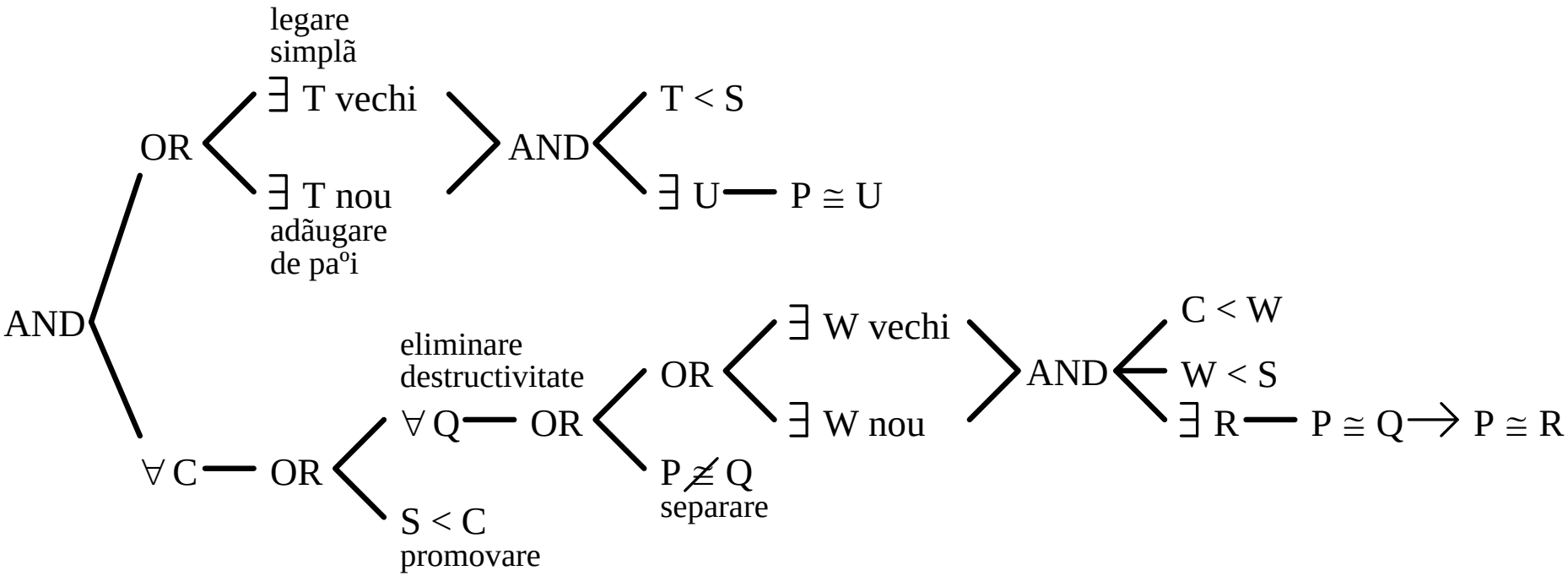
Modelul formal al planificarii - cont

Criteriul adevarului necesar



Modelul formal al planificarii - cont

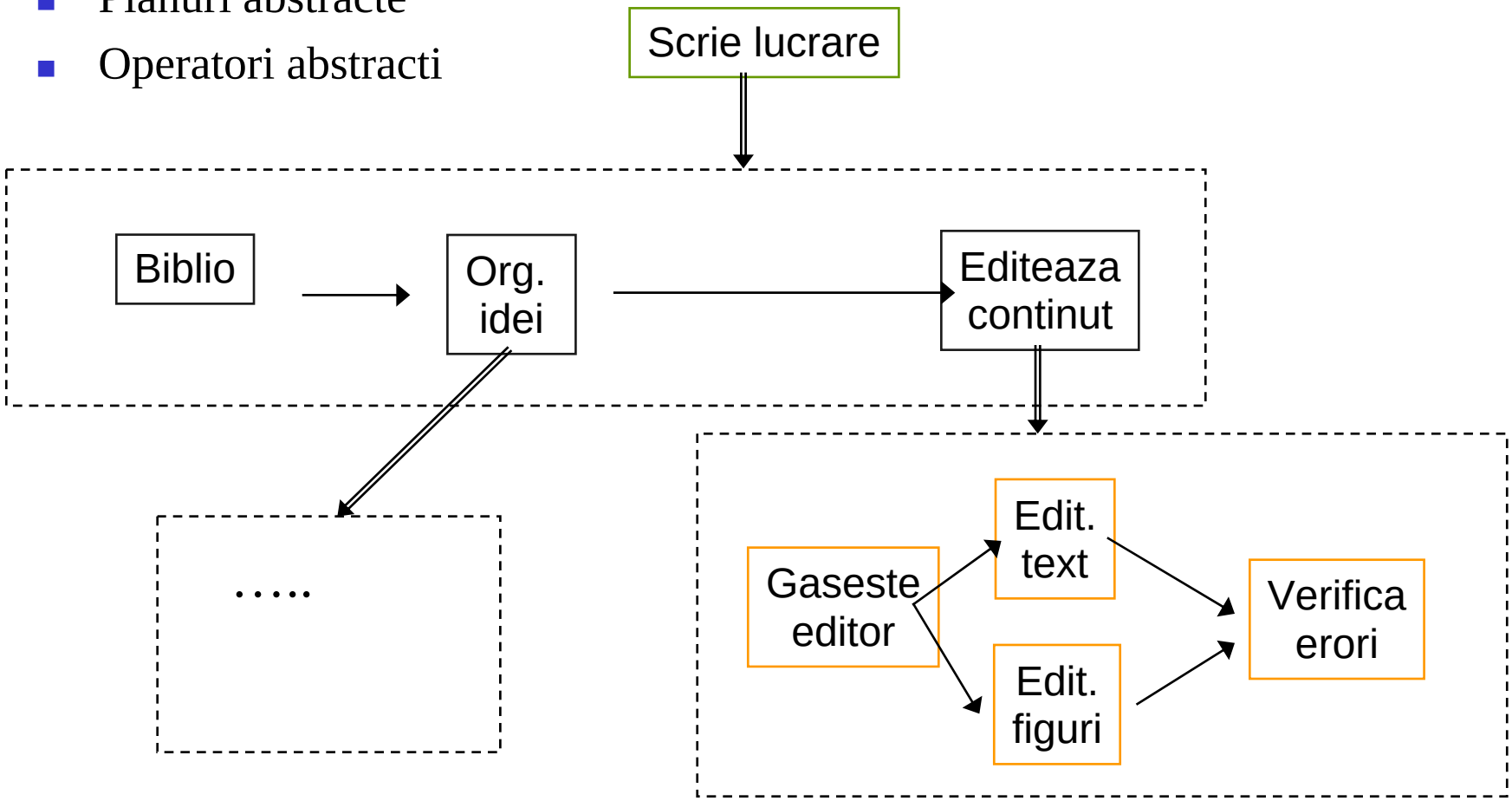
Criteriului adevarului necesar cu operatiile de modificare a planului



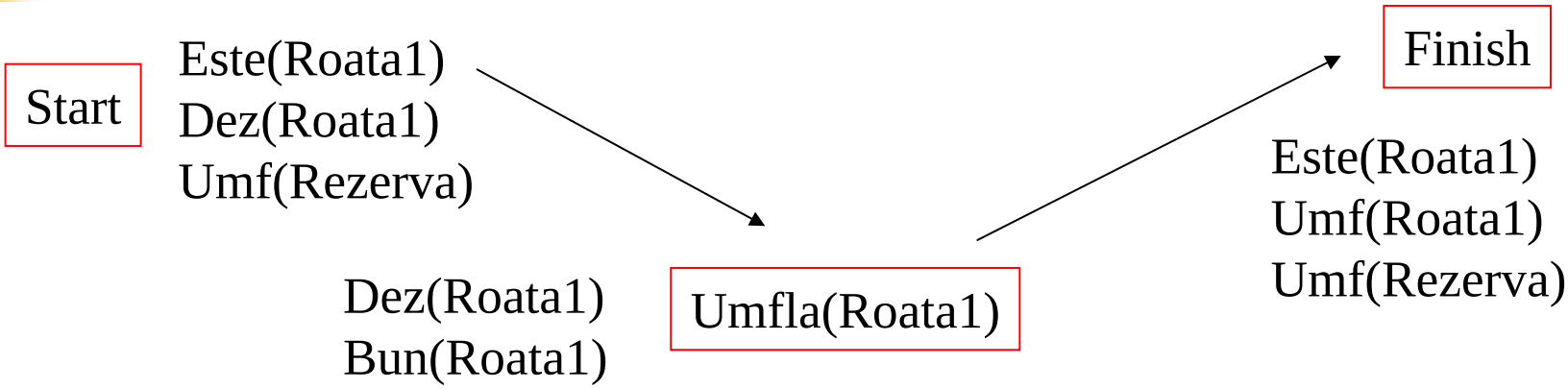
5. Planificare ierarhica

Planuri pe mai multe niveluri de abstractizare

- Planuri abstracte
- Operatori abstracti



6. Planificare contingenta



Planificare contingenta - cont