

Production Systems ES (2)

(Book and Busse book handout)

CSE 352

Lecture Notes (5)

Professor Anita Wasilewska

Forward Chaining

Data -> Rules -> Goal

Also called DATA DRIVEN, BOTTOM UP, or ANTECEDENT chaining

During the **SELECTION** step of each cycle, the RI is looking for applicable rules by **MATCHING** (unifying) condition part of a rule **with the CURRENT CONTENT** of the DB;

Forward chaining is applied, i.e. the proper rule is **FIRED** and a **new FACT** (action part) is **added** to the DB.

Process **TERMINATES** when the **GOAL is reached**, or **when all possible FACTS are already inferred** from the INITIAL database.

Backward Chaining

Also called **GOAL-DRIVEN** consequent chaining

- The production system ESTABLISHES whether a goal is supported by a given database

Start with the goal

-Applicable RULES are found by matching ACTION parts with the **GOAL**

$C_1 \wedge \dots \wedge C_n \rightarrow \text{GOAL}$

Now the conditional part:

$C_1 \wedge \dots \wedge C_n$ is checked against the DB.

If all are (after matching) in DB, the solution is reached.

If C_i is not in DB, we treat it as a SUBGOAL and repeat.

Backward Chaining (continued)

Goal \rightarrow Fact F

Selected rule (by matching action parts with F)

(R) $C_1 \wedge \dots \wedge C_n \rightarrow F$

1. If all $C_1 \wedge \dots \wedge C_n$ are in DB – End
2. Let C be any of $C_1 \wedge \dots \wedge C_n$ (after substitution).
If $\sim C$ is in DB, (R) can't be used and another rule should be selected
3. Neither C nor $\sim C$ is in DB, then
C is a SUBGOAL and we start over again as with F.
4. If no applicable rules exist, F is not established.
System may need new rules.

Usually, backward chaining is executed as depth-first search.

Backward chaining is used in applications with large data.

Forward chaining might produce too much.

Usually, mixed strategies are used.

Example (Busse book)

Simple rule system, no variables.

Knowledge representation = propositional logic

RULES:

R1: IF the ignition key is on
AND the engine won't start
THEN the starting system (including battery) is faulty

R1 $A \wedge B \rightarrow E$

R2: IF E AND the headlights work
THEN the starter is faulty

R2 $E \wedge C \rightarrow G$

R3: IF E AND $\sim C$
THEN the battery is dead

R3 $E \wedge \sim C \rightarrow I$

Example (continued)

R4: IF the voltage test on the ignition switch shows 1 to 6 volts,
THEN the wiring between the ignition and the solenoid is OK

R4 D → F

R5: IF F
THEN replace the ignition switch

R5 F → H

FACTS in the INITIAL DATABASE:

A: The ignition key is on

B: The engine won't start

C: The headlights work

D: The voltage test on the solenoid shows 1 to 6 volts

^ |-----semantics-----|
|

Syntax (in propositional logic representation): A, B, C, D

Initial DB

IDB = {A, B, C, D}

Rules

R1

$A \wedge B \rightarrow E$

R2

$E \wedge C \rightarrow G$

R3

$E \wedge \sim C \rightarrow I$

R4

$D \rightarrow F$

R5

$F \rightarrow H$

GOAL:

Infer all possible facts from IDB

1. Rules are ordered by number

$R_1 < R_2 < R_3 < R_4 < R_5$

2. And they are scanned by RI in this order and inserted into a queue

STEP 1 Applicable: R1, R4

Queue (front to rear): R1, R4

Conflict Resolution: ORDER (1) and

Fire a rule from the front of the queue (and remove it)

Fire: R1 and add E to the IDB

STEP 2 (second cycle) E: The starting system is faulty is added

- R1 is no longer applicable, since its action would add E, which is already in (new) DB (last in C.R.)

- R2 is applicable

Queue (front to rear): R4, R2

- R3 is not applicable; R4 is applicable (and is in queue); R5 is not applicable.

R4 is FIRED from the FRONT of the queue, removed from the queue and new fact

F: The wiring between the ignition and the solenoid is OK
Is added to the DB , now **DBF= { A, B, C, D, E, F }**

STEP 3 (third cycle)

Queue: R2, R5

R5 is inserted, R2 is FIRED (and removed) and new fact

G: The starter is faulty
Is added to the DB, now **DBF = {A, B, C, D, E, F, G}**

STEP 4 (fourth cycle)

Queue: R5

No new rules are applicable, so R5 is fired and new fact

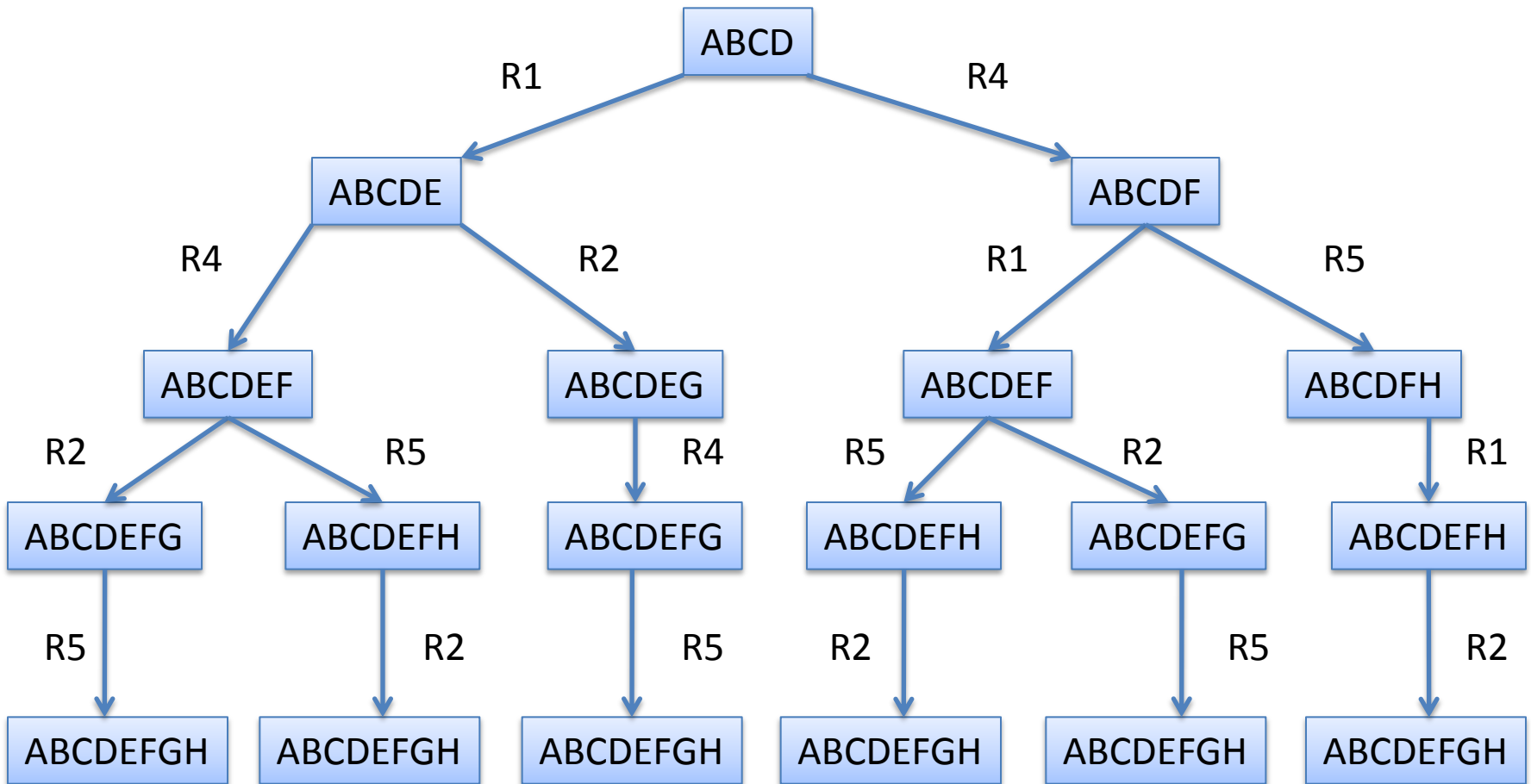
H: Replace the ignition switch
Is added to the DB

STEP 5

No applicable rules (all are used!) **DBF = { A, B, C, D, E, F, G, H }**

RI STOPS COMPUTATION

Search Space



Goal: All possible facts deduced

Initial DB
{A, B, C, D}

GOAL (Using backward chaining):
H∧I

Rules

R1 $A \wedge B \rightarrow E$; **R2** $E \wedge C \rightarrow G$; **R3** $E \wedge \sim C \rightarrow I$;
R4 $D \rightarrow F$; **R5** $F \rightarrow H$

First: Consider H. H is not in the DB. The only rule that matches **H** (action) is

R5: $F \rightarrow H$

Look at F; It is not in the IDB, so **it is a SUBGOAL.** Applicable:

R4 $D \rightarrow F$, and D is in the IDB.

So, **F is SUPPORTED** and hence **H is supported.**

Next: Consider I. I is not in the DB, applicable rule is

R3 $E \wedge \sim C \rightarrow I$

C is in the DB, hence **R3 cannot be used. R3 is the ONLY rule,** hence **I is not supported** and

GOAL H∧I is rejected.

THIS WAS EXAMPLE 2

Example 2 Again

Initial Database: DBF = {A, B, C, D}

Rules

R1: A & B => E

R2: E & C => G

R3: E & ¬ C => I

R4: D => F

R5: F => H

Backward Chaining Goal : H & I

First: Consider H.

H is not in DBF only rule that matches H (as action) is R5.

R5: F => H

Look at F; F is not in DB, so F becomes a subgoal

Applicable: R4: D => F, and D is in DBF so

F is supported and hence H is supported.

Example 2 continued

Next: check I.

I is not in DBF, only applicable rule is **R3: E & ¬ C => I**
C is in DB, hence R3 can't be used.

R3 is the only applicable rule, hence **I is not supported**
and **GOAL H & I is rejected.**

Proportional Logic Conceptualization

Example 3

- R1:** If you are hot, then turn thermostat down
- R2:** If you are not hot and window is open, then close the window
- R3:** If the thermostat is turned down and you are cold, then open the window
- 1. Conceptualize this system in propositional logic**
 - 2. Design questions the program has to ask the user to achieve the goal: “open the window” by backward chaining and conflict resolution**

Example 3 Rules revisited

R1: hot => turn down

R2: \neg hot & window open => close window

R3: thermostat down & cold => open window

GOAL: open window

The GOAL has to be reached by use of conflict resolution and rules R1, R2, R3 from **a certain database of fact.**

We need to build our **DBF** by asking user some questions.

Propositional Logic Conceptualization

H – you are hot

O – window open (open window)

D – Thermostat down

W- close window (closed window)

C- you are cold

R1: $H \Rightarrow D$

R2: $\neg H \ \& \ D \Rightarrow W$

R3: $D \ \& \ C \Rightarrow O$

Goal: reach O by backward chaining

- You need to build your **DBF** by asking questions.

Propositional Logic Conceptualization

Example 3

In order to reach the goal we have only one rule applicable:

R3: $D \ \& \ C \Rightarrow O$

We have two **subgoals: D, C**

We get **D** by **R1: $H \Rightarrow D$** and D becomes a **subgoal**.

No applicable rule, so we need ask a question about **H**.

Question: Are you hot (H) ?

If answer is **YES**: we **ADD H** into DBF , i.e.

DBF = {H} and we apply (fire) **R1: $H \Rightarrow D$** and get **D**.

D is supported

If answer is **NO**, we added $\{\neg H\}$ to DBF, i.e **DBF = $\{\neg H\}$** .

No applicable rule, **we STOP**.

We look for **C**, **no applicable rule**.

GOAL O IS REJECTED.

OBSERVATION:

FACTS are always true in ES Database

For example a Fact:

(car#42, battery, weak)

means that in our database we have a record

Key	Other attribute	Other attribute	Battery		
Car#42			weak		

Example 4

Key	Other attribute	Other attribute	Battery		
Car#42			weak		

Another way of writing the fact **(car#42, battery, weak)** is:

Battery(car#42, weak)

This is called a predicate form.

Atomic formula written in a **triple form** is:

(x, battery, weak)

Atomic formula written in a **predicate form** is:

battery(x, weak)

Example 5: given a DB

Cars	Battery	Color	Buy	Garage
C ₁	Good	Red	Yes	2
C ₂	Weak	Black	No	1

Rules in our ES (in a triple form) are:

**R1. (x, battery, good) & (color, red) =>
(x, garage, 2)**

R2. (x, battery, weak) & (x, gar, 1) =>(x, buy, no)

Observe that the rules R1, R2 are true in our DB!

Example 5 (continued)

Facts in our ES are:

F1. (C_1 , battery, good)

F2. (C_1 , color, red)

F3. (C_2 , battery, weak)

F4. (C_2 , garage, 1)

Question 1: What our ES with R1 and R2 will deduce from these facts?

Example 5 (continued)

FACTS written in the predicate form are:

F1. battery(C_1 , good)

F2. color(C_1 , red)

F3. battery(C_2 , weak)

F4. garage(C_2 , 1)

Question 2: write rules R1 and R2 in predicate form and use conflict resolution to deduce all you can from facts F1-F4.