



# Introduction to Production Systems

Course: Cognitive Systems 2  
Winter 2008/09



# Overview

- What is a production system (PS)?
  - Rules
  - Data base
  - Interpreter
- Basic characteristics of PS
- Appropriate domains for PS models
- PS taxonomy



# What is a production system?

- Production systems are programming languages
- Widely employed for representing the processes in models of cognitive systems
- Instructions (i.e., the productions) are of the form  
IF <conditions>, THEN <actions>  
or  
 $C \rightarrow A$ , for short



# Recall formal languages and grammars

$$A = \{a,b,c\}$$

$$L_1(A) = \{a^n b^n c^n \mid n \in \mathbb{N}^+\}$$

$$G = \{N,T,P,S\}, N = \{S,B\}, T = \{a,b,c\}$$

$$P = \{S \rightarrow aBSc, S \rightarrow abc, Ba \rightarrow aB, Bb \rightarrow bb\}$$

$$S \Rightarrow abc$$

$$S \Rightarrow aBSc \Rightarrow aBabcc \Rightarrow aaBbcc \Rightarrow aabbcc$$

...



# Recall Chomsky hierarchy

- 3: Regular grammars and languages, finite automata.  $A \rightarrow a$  and all other rules either  $A \rightarrow aB$  or  $A \rightarrow Ba$ .  $a^n$
- 2: Context-free grammars and languages, nondeterministic pushdown automata.  $A \rightarrow \gamma$ .  $a^n b^n$
- 1: Context-sensitive grammars and languages, linear bounded automata.  $\alpha A \beta \rightarrow \alpha \gamma \beta$ .  $a^n b^n c^n$
- 0: Unrestricted grammars, recursively enumerable languages, Turing machine.  $\alpha \rightarrow \beta$ .



# Production systems

- Productions  $C \rightarrow A$  are (usually) unrestricted.
- Turing complete
- Components: Rules, database (DB), interpreter
- Rules are matched against DB content
- All rules that match are executed; order depends on preferences
- Computation is done when all no rules match anymore

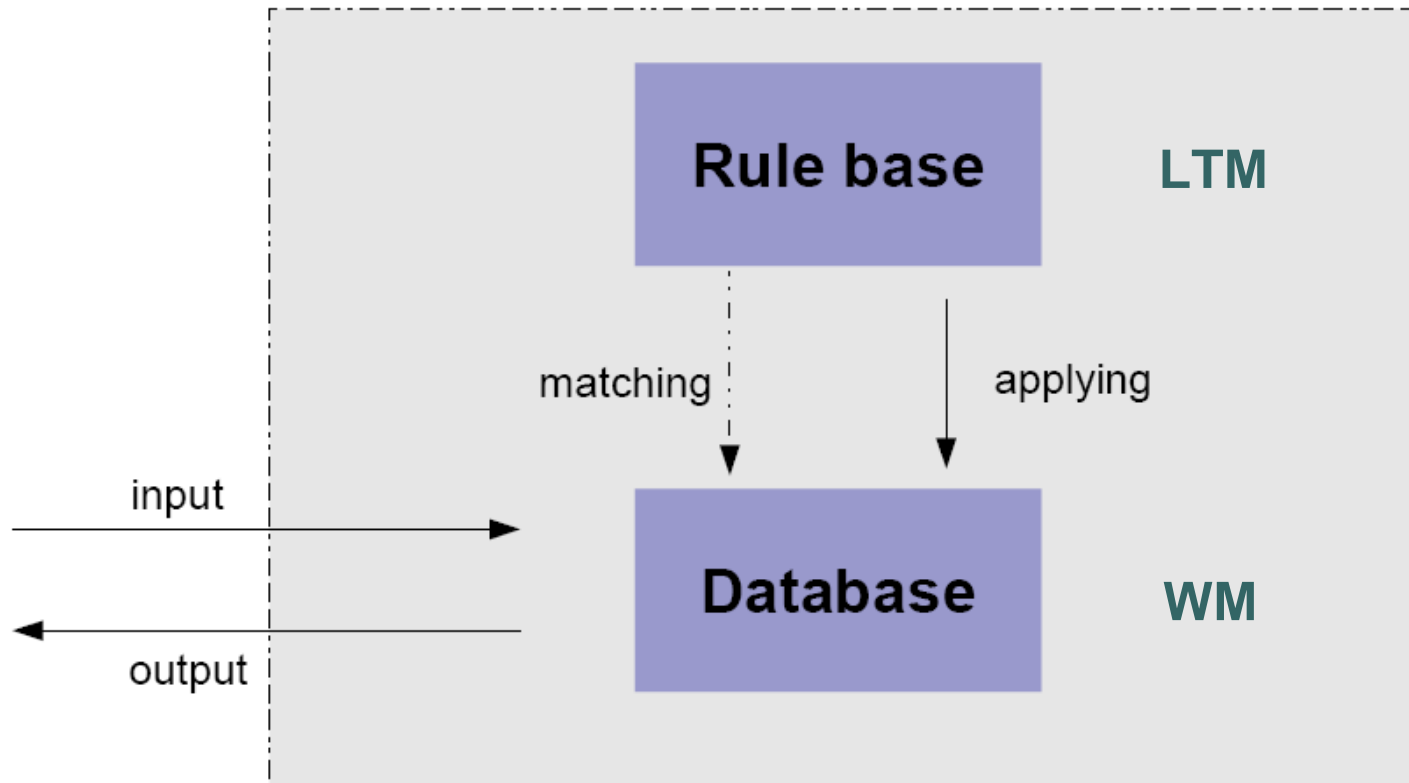


## Rule execution $C \rightarrow A$

- Rule set has predetermined total ordering
- DB is collection of symbols (reflects the state of affairs in the computation)
- Interpreter scans left hand sides of rules (LHS) until it finds one that it can match against DB.
- Upon rule execution, matching symbols in LHS of rule are replaced in DB with those in RHS of rule. Scanning is then resumed.
- Basic control flow: Every rule may match at any time.



# Production systems





# Rules

- Rules are often seen to represent procedural knowledge
- They relate to classical stimulus-response connections in psychology ( $S \rightarrow R$ ) but are more flexible than these as both LHS and RHS may contain variables.
- Matching variants:
  - Match LHS (forward chaining):  
produce expressions
  - Match RHS (backward chaining, goal-driven):  
recognize expressions









# Rules

- More generally: one side of rule is evaluated with reference to the DB.
- Simplest evaluation: matching of literals.
- Simplest action: replacement.
- Conflicts arise when several rules may be executed. Ordering of rules in terms of precedence needed (either explicitly or implicitly defined).
- Conflict resolution: process of selecting a rule

● ● ● | An example

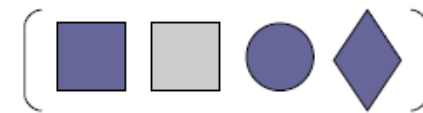
PS = (P1 , P2 , p3)

- P1 :=  → 
- P2 :=  → 
- P3 :=  → 

Control cycle :

Rule selection : P1 > P2 > P3  
Left-Hand-Side matching







Initial Database :



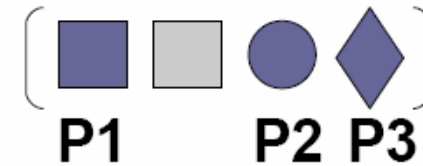
# ● ● ● | An example

## Matching

Rules matching :

- P1 :=  → 
- P2 :=  → 
- P3 :=  → 

Current database :



*Matching rules (Conflict set) : { (P1,1) ,(P2,3) (P3,4) }*



# An example

## Conflict resolution

Rule selection : **P1** > **P2** > P3  
Left-Hand-Side matching



Rule **P1** will be activated




# An example

## Applying rule

Selected rule




•  $P1 :=$    $\longrightarrow$  

DB before applying:

(     )

Performing action





Database now :

(     )

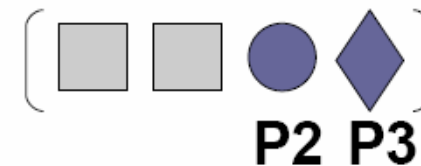
● ● ● | An example

## Matching

Rules matching :

- P2 :=  → 
- P3 :=  → 

Current database :



*Matching rules (Conflict set) : { (P2,3) (P3,4) }*



# An example

## Conflict resolution

Rule selection : P1 > **P2** > P3  
Left-Hand-Side matching



Rule **P2** will be activated

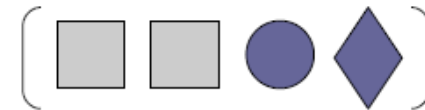
● ● ● | An example

applying rule

Selected rule

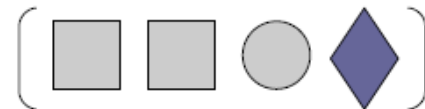
•  $P2 := \text{blue circle} \rightarrow \text{grey circle}$

DB before applying:



Performing action

Database now :



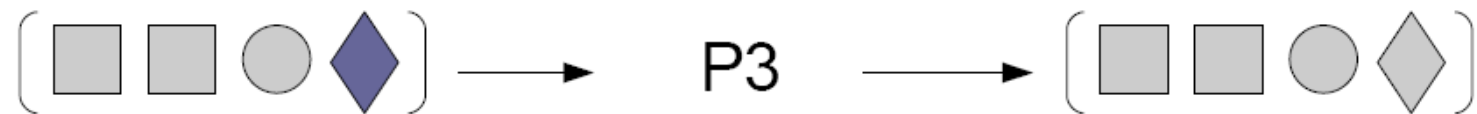
● ● ● | An example

matching :

• P3 :=  → 

Rule chosen : **P3**

act :



No more rules do match , we are done



# Data base

- Sole storage medium for state of system
- For purposes of human cognitive modeling:
  - Size of data base may be related to storage capacities of human memory (WM / STM)
  - Important dimension: number of knowledge chunks held at one time
- For expert systems:
  - Contains facts & assertions about the world



# Unity of data and control store

- PS do only store information in the DB and the rule base
- No separate storage of control state information (program counters, stacks, ...)
- DB is universally accessible to every rule at every evaluation step
- Control flow is determined by interplay between rules and DB content over time and by implemented conflict resolution mechanisms



# Interpreter

- Select-execute loop
  - Choose applicable rule
    - Select applicable rules (conflict set)
    - Choose from set (conflict resolution)
  - Apply rule
- Alternation between selection and execution is essential characteristic of PS
- Total reevaluation of entire PS at each step
  - global scope
  - Flexible adaptation to changing environment
  - Much computation required



# Conflict resolution strategies

- Rule order: Every rule has a priority. Choose rule with highest priority.
- Data order: Elements of DB are ordered. Choose rule that matches element with highest priority.
- Generality order: Most specific rule is chosen (LISP70)
- Rule precedence: A precedence network determines the hierarchy. (DENDRAL)
- Recency order: Either the most recently executed rule or the rule for the most recently updated element in DB is chosen.
- Meta Rules: There are rules in the PS to select the rules (goal orientated, e.g. MYCIN)
- Random choice
- ...



# PS types

- Historically, production systems were chiefly used for two purposes:
  - Psychological / cognitive modeling
  - Technical applications (knowledge-based expert systems)



# Psychological / cognitive modeling

- Program should embody a theory of how humans perform a task
- Rules are written so that system is able to reproduce human behavior
- Behavior is meant to include errors, forgetting, shortcomings typical of human cognition
- Goal: a better understanding of human cognitive architecture
- PS may lead to new hypotheses on human cognition

→ In focus in this course



# Technical applications / expert systems

- Performance-oriented (no errors wanted!)
- Usually large knowledge bases
- Design advantages of PS:
  - Decompositional / incremental
  - Basic components are easy to change
  - Simple, direct



# Basic characteristics of PS

- Each rule can fire at any time
  - Each rule could fire if conditions are satisfied
  - Hard to debug if not designed properly
  - Powerful to cope with “strange” situations
- No direct interaction between rules
  - Design new DB entries when necessary
  - Control elements
    - Symbols that are understood by just one rule
    - Symbols that exclude rules from match
  - Adequacy for purposes of cognitive modeling?



# Basic characteristics of PS

- (Traditionally) Constrained format
  - Match & manipulate as primitives
  - Simple matching allows partial checks for failure
  - Machine readability
    - Automatic consistency checking
    - Automatic rule generation / deletion / alteration
  - Meta-rules
    - Heuristics for conflict resolution
    - Information about reasoning / actions
    - ...



# Basic characteristics of PS

- High modularity
  - Easy to change or add single functionalities
  - Easy „damaging“ for experimental purposes





# Appropriate domains for PS models

Approach favors problems

- With many independent states
  - Multiple, nontrivially different, independent states → multiple, nontrivial, modular rules
- With limited communication between actions (independent actions rather than coordinated parallel processing)
- With limited explicit descriptions of control flow
- In which knowledge can be kept rather separate from the manner in which it is to be used (declarative vs. procedural knowledge)



# PS Taxonomy

- Form: How simple or complex should the syntax of each side be?
  - Simple rules
    - Simple matching conditions (e.g. literals)
    - Simply structured actions (e.g. replacement)
  - More complex rules
    - Complex matching conditions (# of free variables, node networks & spreading activation, complex data structures, ...)
    - Complex actions ('programs') to be executed



# PS Taxonomy

- Knowledge content
  - Which knowledge should go into rules?
  - Level of conceptual primitives (simple string replacements vs. complex reasoning steps)
- Control cycle architecture
  - LHS vs. RHS: role of goal-driven problem solving
  - Selection of conflict resolution strategies



# PS Taxonomy

- System extensibility
  - Automatic bug tracking
    - Modes of bug fixing (automatic / manual)
    - Predictability of outcome of bug fixing
  - Learning as augmentation of a system's rule base
    - Automatic generation of new rules
    - Automatic classification / detailing of new rules
    - How difficult is it to change control structure information? Stored in rules or higher-order rules?
    - How is consistency of the rule base assured? Is it?
  - ...



# Conclusions

- PS are general, powerful programming systems
- They provide a powerful basis for modeling
- Well suited for domains with many knowledge pieces
- Suitable for cognitive modeling
- Rules can be treated as procedural or declarative knowledge
- Restricted syntax and non-explicit control flow require systematic, well disciplined software development
- Comparably high temporal complexity due to the matching of all rules in each step



# Sources

- Davis, R. and King J.J. The origin of rule-based systems in AI. (see course reader)
- Simon, H.A. (1999) Production systems. In R. A. Wilson and F. C. Keil: *The MIT Encyclopedia of Cognitive Sciences*. Cambridge, MA: MIT Press.
- Simon, H. A. (1996). *The Sciences of the Artificial*. 3rd ed. Cambridge, MA: MIT Press.
- Various presentations from previous semesters

---

Sven Bertel (slides), Thomas Barkowsky, Denise Peters,  
Christian Freksa