Rule Systems

CPSC 433: Artificial Intelligence
Fall 2024

Jonathan Hudson, Ph.D.
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

August 8, 2024

UNIVERSITY OF CALGARY

Rule Systems

Rule Sets / Production Systems

- Focus on operational knowledge:

  if condition then action

- Uses a logic:

  usually propositional or multi-valued

  - probabilistic rules

- Actions include input-requests, output, changes of knowledge-base
- If several rules can be applied, a conflict manager decides what to do

  ☞ defines the operational semantics of the system

  ☞ must be well understood by knowledge engineer

Rule Sets / Production Systems

- Focus on operational knowledge:
  if condition then action

- Actions include input-requests, output, changes of knowledge-base

- If several rules can be applied, a conflict manager decides what to do
  - defines the operational semantics of the system
  - must be well understood by knowledge engineer

PROLOG

PROLOG

General idea:

1. program descriptively by just stating axioms in a logic and asking queries
2. guide interpreter by clear evaluation control scheme
3. whole concept is based on SLD-resolution

Basic data structures

1. Horn clauses in first-order logic,
   - i.e. clauses of form

     $\neg \boldsymbol{A_1} \vee \neg \boldsymbol{A_2} \vee \cdots \vee \neg \boldsymbol{A_n} \vee \boldsymbol{B}$
   - written:

     $\boldsymbol{B} :- \boldsymbol{A_1}, \boldsymbol{A_2}, \ldots, \boldsymbol{A_n}$
   - read:

     if $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$and … and $\boldsymbol{A_n}$ then $\boldsymbol{B}$

     $\boldsymbol{A_1} \wedge \boldsymbol{A_2} \wedge \cdots \wedge \boldsymbol{A_n} \rightarrow \boldsymbol{B}$

2. Some higher-order predicates to manipulate the set (list) of clauses in the knowledge base, influence the interpreter, or provide in- and output.

Database

- We generally have a data base/knowledge base:

$$
\begin{array}{l}
\text{Data base:} \\
\quad \dots \\
A_1 :- B_{11}, \dots, b_{1n_1}. \\
\quad \dots \\
A_k :- B_{k1}, \dots, b_{kn_k}. \\
\dots
\end{array}
$$

- This is a list of horn clauses in Prolog form, the left side (knowledge A) exists for each, if all of the right side (knowledge Bs) exist
- Notice prolog clauses use :- as divider and a period . at the end

Goal Stack

- We generally have a goal stack (or a query):
- This is something we are trying to see if it fits with the previous database on knowledge (here we have m sub-goals in our query/goal stack)

$$?-G_1, G_2, \ldots, G_m$$

- We basically will move forward trying to solve the goals by examining the data we have in our world (remember that we will have predicates in prolog and this means we have to worry about unification (mgu) being consistent during process)

Semantics

- Operational semantic using a goal stack and the list of clauses (data base/knowledge base):

$$\begin{tikzcd} ?-G_1, G_2, \dots, G_m \arrow[d, "\sigma_1(G_1) = \sigma_1(A_1)"] \\ ?-\sigma_1(B_{11}), \dots, \sigma_1(B_{1n_1}), \sigma_1(G_2), \dots, \sigma_1(G_m) \end{tikzcd}$$

Data base:

$\dots$

$\boldsymbol{A_1 :- B_{11}, \dots, b_{1n_1}.}$

$\dots$

$A_k :- B_{k1}, \dots, b_{kn_k}.$

$\dots$

- We replace our goal by unifying against
  - Our first piece of database knowledge
  - Which adds those pieces of knowledge to the goal stack/query

σ_1 mgus

Semantics

- Operational semantic using a goal stack and the list of clauses (data base/knowledge base):

$$\begin{tikzcd} ?-G_1, G_2, \dots, G_m \arrow[d, "\sigma_1(G_1) = \sigma_1(A_1)"] \\ ?-\sigma_1(B_{11}), \dots, \sigma_1(B_{1n_1}), \sigma_1(G_2), \dots, \sigma_1(G_m) \end{tikzcd}$$

$\sigma_1(B_{ij})$ not solvable?

backtrack

Data base:

…

$A_1 :- B_{11}, \dots, b_{1n_1}.$

…

$A_k :- B_{k1}, \dots, b_{kn_k}.$

…

σ_1 mgus

Semantics

- Operational semantic using a goal stack and the list of clauses (data base/knowledge base):

$$\begin{tikzcd} ?-G_1, G_2, \dots, G_m \arrow[d, "\sigma_2(G_1) = \sigma_2(A_2)"] \\ ?-\sigma_2(B_{21}), \dots, \sigma_2\left(B_{2n_2}\right), \sigma_2(G_2), \dots, \sigma_2(G_m) \end{tikzcd}$$

Data base:

$\dots$

$A_1 :- B_{11}, \dots, b_{1n_1}.$

$A_2 :- B_{21}, \dots, b_{2n_2}.$

$\dots$

$A_k :- B_{k1}, \dots, b_{kn_k}.$

$\dots$

σ_2 mgus

Semantics (II)

1. Solution: if goal stacks get empty
   - collect substitutions that fulfilled original goals
   - use as answer

2. Next solution (for more than one fulfilling answer): initiate backtrack

3. No solution:
   - if no clause in data base solves a particular subgoal G_i for all solutions to $G_1,\ldots,G_{i-1}$

How to get knowledge into the representation structure

Getting knowledge into system?

- By writing a declarative problem description

- Caution: take into account the semantics!

    - Especially that we have an and-or-tree-based search with a special depth-first control
        - (that in fact boils down to and-tree-based search with backtracking)
        - ordering of clauses in data base very important:
            - from very specialized to very general

Discussion

- In theory: describing knowledge by logic rules enough; no control necessary
- Fast prototyping very easy!

- Not really much left from logic
- Exact understanding of operational semantic necessary to use
    - ☞Just another (not very efficient) programming language if you don’t

And what about processing data?

- Follow operational semantics
  - not really search

- Rely on user/programmer knowing what he/she is doing

PROLOG Examples

Examples

- Write a PROLOG program that given facts of the form

  ```
  mother(X,Y).        father(X,Y).
  ```

  meaning a is mother, resp. father of b, answers questions like

  ```
  ?- grandmother(agnes,X).
  ?- grandfather(Y,clara).
  ```

- Home exercise: Given the facts:

  ```
  mother(anna,peter). mother(anna,clara). father(joe,peter). father(jim,clara).
  mother(mary,anna). father(tom,joe).
  ```

  Answer:

  ```
  ?- grandfather(tom,X).
  ?- grandmother(X,peter).
  ```

Examples

mother(X,Y).

father(X,Y).

Note these two are not actually rules we would store but reminding you of that we plan to use to form grandmother/grandfather

Examples

```
mother(X,Y).
father(X,Y).
grandmother(X,Y):-mother(X,Z),mother(Z,Y).
```

Examples

```
mother(X,Y).
father(X,Y).
grandmother(X,Y):-mother(X,Z),mother(Z,Y).
grandfather(X,Y):-father(X,Z),father(Z,Y).
```

Examples 1

```
mother(X,Y).
father(X,Y).
grandmother(X,Y):-mother(X,Z),mother(Z,Y).
grandfather(X,Y):-father(X,Z),father(Z,Y).
grandmother(X,Y):-mother(X,Z),father(Z,Y).
grandfather(X,Y):-father(X,Z),mother(Z,Y).
```

Examples 2

```
mother(X,Y).
father(X,Y).
parent(X,Y):-mother(X,Y).
parent(X,Y):-father(X,Y).
grandmother(X,Y):-mother(X,Z),parent(Z,Y).
grandfather(X,Y):-father(X,Z),parent(Z,Y).
```

Examples 2 - Database

```
parent(X,Y):-mother(X,Y).
parent(X,Y):-father(X,Y).
grandmother(X,Y):-mother(X,Z),parent(Z,Y).
grandfather(X,Y):-father(X,Z),parent(Z,Y).
mother(anna,peter).
mother(anna,clara).
father(joe,peter).
father(jim,clara).
mother(mary,anna).
father(tom,joe).
```

Examples 2 – Goal 1

parent(X,Y):-mother(X,Y).

parent(X,Y):-father(X,Y).

grandmother(X,Y):-mother(X,Z),parent(Z,Y).

grandfather(X,Y):-father(X,Z),parent(Z,Y).

mother(anna,peter).

mother(anna,clara).

father(joe,peter).

father(jim,clara).

mother(mary,anna).

father(tom,joe).

```
?-grandmother(X,peter).
X=mary
```

Examples 2 – Goal 2

```
parent(X,Y):-mother(X,Y).
parent(X,Y):-father(X,Y).
grandmother(X,Y):-mother(X,Z),parent(Z,Y).
grandfather(X,Y):-father(X,Z),parent(Z,Y).
mother(anna,peter).
mother(anna,clara).
father(joe,peter).
father(jim,clara).
mother(mary,anna).
father(tom,joe).
```

```
?-grandfather(tom,X).
X=peter
```

https://swish.swi-prolog.org/

Examples 2 – Goal 1

```
parent(X,Y):-mother(X,Y).
parent(X,Y):-father(X,Y).
grandmother(X,Y):-mother(X,Z),parent(Z,Y).
grandfather(X,Y):-father(X,Z),parent(Z,Y).
mother(anna,peter).
mother(anna,clara).
father(joe,peter).
father(jim,clara).
mother(mary,anna).
father(tom,joe).
```

```
?-grandmother(X,peter).
?-mother(X,Z),parent(Z,Y). {Y/peter}
?-parent(Z,Y). {Y/peter,X/anna,Z/peter}
?-mother(Z,Y). {Y/peter,X/anna,Z/peter}
?-father(Z,Y). {Y/peter,X/anna,Z/peter}
?-parent(Z,Y). {Y/peter,X/anna,Z/clara}
?-mother(Z,Y). {Y/peter,X/anna,Z/clara}
?-father(Z,Y). {Y/peter,X/anna,Z/clara}
?-parent(Z,Y). {Y/peter,X/mary,Z/anna}
?-mother(Z,Y). {Y/peter,X/mary,Z/anna}
?-■

X=mary
```

MYCIN/EMYCIN

MYCIN / EMYCIN

General ideas:

- Deal with unsure/uncertain knowledge
- Use in expert system
  ☞ dialog with user
- MYCIN: medical expert system
- EMYCIN: expert system shell employing logic, semantics, calculus and control of MYCIN, not the particular knowledge

Basic data structures

- Object-attribute-value triples as base logic:
  for all a $\in$ F: τ(a) = 0 (objects and values)
  for all x $\in$ V: τ(x) = 0 (object and value variables)
  for all A $\in$ PI: τ(A) = 2 (attributes)
  P = PV = {}

Basic data structures

- Object-attribute-value **triples** as base logic:
- realized as A(e,v):
  - attribute A of object e has value v

- Production rules form the formulas:
  $J = \{\neg,\wedge\} \cup \{\rightarrow_i | i = 1,\ldots,n$, if there are n production rules$\}$
  $Q = \{\}$

- Deal with uncertain knowledge by using W=[-1,1]
  - (resp. discrete representation {-1,-0.9,…,0.9,1})

Semantics

- Interpret all symbols for a fixed domain D
- Start with given interpretation for selected object-attribute-value-triples (input-data) and given truth values for all production rules
- Use operational semantics based on computing
  - Measure of belief (MB)
  - Measure of disbelief (MD)

Measures of belief/disbelief

Let h be an object-attribute-value triple and e a set of production rules.

If $P_1 \wedge \ldots \wedge P_n \rightarrow_i h$ is the only rule in e, then we get

MB(h,e) =

$$I(P_1 \wedge \ldots \wedge P_n \rightarrow_i h) * \max(0, \min(I(P_1), \ldots, I(P_n)))$$

Measures of belief/disbelief

If $e = \{e_1,e_2\}$, then we get

$MB(h,\{e_1,e_2\}) = 0$, if $MD(h,\{e_1,e_2\}) = 1$

$MB(h,\{e_1,e_2\}) = MB(h,\{e_1\}) + MB(h,\{e_2\}) * (1-MB(h,\{e_1\}))$

Measures of belief/disbelief

If $e = \{e_1,e_2\}$, then we get

$MB(h,\{e_1,e_2\}) = 0$, if $MD(h,\{e_1,e_2\}) = 1$

$MB(h,\{e_1,e_2\}) = MB(h,\{e_1\}) + MB(h,\{e_2\}) * (1-MB(h,\{e_1\}))$

For more elements just iterate this.

MD is computed similarily, except that e contains all production rules of the form $P_1 \wedge \ldots \wedge P_n \rightarrow_i \neg h$

☞ application of Bayes formula for conditional probabilities

How to get knowledge into the representation structure

- The rules are defined by an expert, who also defines what objects and attributes are of interest and what values they can have.
- The expert also has to provide the interpretation for the rules, by expressing how confident he/she is in this rule
- The interpretation for the input data is provided by observation/measuring of the world (in MYCIN, by a doctor interpreting the examinations of the patient).

Discussion

- Allows to deal with uncertainty
- Successful in several applications

- Application domain has to be small
- Hands-on approach to probability theory
- Hides the need for TMS (Truth Maintenance System)
- Gets very complicated for large rule sets with the same conclusion

And what about processing data?

- Very similar to PROLOG
- Rules are applied backwards:
  - Select an object-attribute-value-triple for which an interpretation is sought and add it to the goal list:
  - Repeat:
    - Select h from goal list
    - Find a rule e with h or $\neg$h as consequence
    - Add premises to goal list and update interpretation of h by MB(h,e) - MD(h,e), resp. combine values for h from other rules

MYCIN/EMYCIN Examples

Examples (I)

- Construct MYCIN rules for the following knowledge:
  - If the preparation for the exam is good and the student slept well, then there is a good chance (0.7) that the student will pass the exam.
  - If the student's contribution to the team effort is high and the workload of the student is low, then there is a good chance (0.8) that the student will pass the exam.
  - If the workload of the student is high and the extra-curricular activities are high, then there is a good chance (0.6) that the student will fail the exam.

Examples (I)

- Construct MYCIN rules for the following knowledge:
    - If the preparation for the exam is good and the student slept well, then there is a good chance (0.7) that the student will pass the exam.

Examples (I)

- Construct MYCIN rules for the following knowledge:
  - If the preparation for the exam is good and the student slept well, then there is a good chance (0.7) that the student will pass the exam.
  - $e_1 = prep(X, good) \wedge slept(X, well) \rightarrow_1 exam(X, pass) \qquad I(e_1) = 0.7$

Examples (I)

- Construct MYCIN rules for the following knowledge:
  - If the preparation for the exam is good and the student slept well, then there is a good chance (0.7) that the student will pass the exam.
  - $e_1 = prep(X, good) \land slept(X, well) \rightarrow_1 exam(X, pass) \qquad I(e_1) = 0.7$
  - If the student's contribution to the team effort is high and the workload of the student is low, then there is a good chance (0.8) that the student will pass the exam.
  - $e_2 = contr(X, high) \land work(X, low) \rightarrow_2 exam(X, pass) \qquad I(e_2) = 0.8$

Examples (I)

- Construct MYCIN rules for the following knowledge:
  - If the preparation for the exam is good and the student slept well, then there is a good chance (0.7) that the student will pass the exam.
  - $e_1 = prep(X, good) \wedge slept(X, well) \rightarrow_1 exam(X, pass) \qquad I(e_1) = 0.7$
  - If the student's contribution to the team effort is high and the workload of the student is low, then there is a good chance (0.8) that the student will pass the exam.
  - $e_2 = contr(X, high) \wedge work(X, low) \rightarrow_2 exam(X, pass) \qquad I(e_2) = 0.8$
  - If the workload of the student is high and the extra-curricular activities are high, then there is a good chance (0.6) that the student will fail the exam.
  - $e_3 = work(X, high) \wedge extra(X, high) \rightarrow_3 \neg exam(X, pass) \qquad I(e_3) = 0.6$

Examples (I)

- Construct MYCIN rules for the following knowledge:
  - If the preparation for the exam is good and the student slept well, then there is a good chance (0.7) that the student will pass the exam.
  - $e_1 = prep(X, good) \land slept(X, well) \rightarrow_1 exam(X, pass) \qquad I(e_1) = 0.7$
  - If the student's contribution to the team effort is high and the workload of the student is low, then there is a good chance (0.8) that the student will pass the exam.
  - $e_2 = contr(X, high) \land work(X, low) \rightarrow_2 exam(X, pass) \qquad I(e_2) = 0.8$
  - If the workload of the student is high and the extra-curricular activities are high, then there is a good chance (0.6) that the student will fail the exam.
  - $e_3 = work(X, high) \land extra(X, high) \rightarrow_3 \neg\, exam(X, pass) \qquad I(e_3) = 0.6$

Examples (II)

- Interpret the statement
  Joe passes the exam
  if you know that
  - I(prep(Joe,good)) = 0.7
  - I(sleep(Joe, well)) = 0.6
  - I(contr(Joe,high)) = 0.9
  - I(work(Joe,low)) = 0.6
  - I(extra(Joe,high)) = 0.3

Examples (II)

- Interpret the statement

  Joe passes the exam

  I(exam(Joe,pass)) = ?

- if you know that
  - I(prep(Joe,good)) = 0.7
  - I(sleep(Joe, well)) = 0.6
  - I(contr(Joe,high)) = 0.9
  - I(work(Joe,low)) = 0.6
  - I(extra(Joe,high)) = 0.3

Examples (II)

- Interpret the statement

  Joe passes the exam

  I(exam(Joe,pass)) = ?

  $MB(exam(Joe, pass), \{e_1, e_2\}) - MD(exam(Joe, pass), \{e_3\})$

- if you know that
  - I(prep(Joe,good)) = 0.7
  - I(sleep(Joe, well)) = 0.6
  - I(contr(Joe,high)) = 0.9
  - I(work(Joe,low)) = 0.6
  - I(extra(Joe,high)) = 0.3

Examples (II)

- Interpret the statement

  Joe passes the exam

  I(exam(Joe,pass)) = ?

  $MB(exam(Joe, pass), \{e_1, e_2\}) - MD(exam(Joe, pass), \{e_3\})$

  $MB(exam(Joe, pass), e_1) + MB(exam(Joe, pass), e_2) * (1 - MB(exam(Joe, pass), e_1))$
  $- MD(exam(Joe, pass), \{e_3\}$

- if you know that
  - I(prep(Joe,good)) = 0.7
  - I(sleep(Joe, well)) = 0.6
  - I(contr(Joe,high)) = 0.9
  - I(work(Joe,low)) = 0.6
  - I(extra(Joe,high)) = 0.3

Examples (II)

- Interpret the statement

  Joe passes the exam

  I(exam(Joe,pass)) = ?

  $MB(exam(Joe,pass),\{e_1, e_2\}) - MD(exam(Joe,pass),\{e_3\})$

  $MB(exam(Joe,pass),e_1) + MB(exam(Joe,pass),e_2) * (1 - MB(exam(Joe,pass),e_1))$
  $- MD(exam(Joe,pass),\{e_3\}$

  $MB(h,e) = I(P_1 \wedge \ldots \wedge P_n \rightarrow_i h) * \max(0,\min(I(P_1),\ldots,I(P_n))$

  $e_1 = prep(X,good) \wedge slept(X,well) \rightarrow_1 exam(X,pass)$ $I(e_1) = 0.7$
  $MB(exam(Joe,pass),e_1) = 0.7 * \max(0,\min(0.7,0.6)) = 0.42$

- if you know that
  - I(prep(Joe,good)) = 0.7
  - I(sleep(Joe, well)) = 0.6
  - I(contr(Joe,high)) = 0.9
  - I(work(Joe,low)) = 0.6
  - I(extra(Joe,high)) = 0.3

Examples (II)

- Interpret the statement

  Joe passes the exam

  I(exam(Joe,pass)) = ?

  $MB(exam(Joe,pass),\{e_1\ ,e_2\}) - MD(exam(Joe,pass),\{e_3\})$

  $0.42 + MB(exam(Joe,pass),e_2) * 0.58 - MD(exam(Joe,pass),\{e_3\}$

  $MB(h,e) = I(P_1 \wedge \ldots \wedge P_n \rightarrow_i h) * \max(0,\min(I(P_1),\ldots,I(P_n)))$

  - $e_2 = contr(X,high) \wedge work(X,low) \rightarrow_2 exam(X,pass)$ $I(e_2) = 0.8$

    $MB(exam(Joe,pass),e_2) = 0.8 * \max(0,\min(0.9,0.6)) = 0.48$

- if you know that
  - I(prep(Joe,good)) = 0.7
  - I(sleep(Joe, well)) = 0.6
  - I(contr(Joe,high)) = 0.9
  - I(work(Joe,low)) = 0.6
  - I(extra(Joe,high)) = 0.3

Examples (II)

- Interpret the statement

  Joe passes the exam

  I(exam(Joe,pass)) = ?

  $MB(exam(Joe, pass), \{e_1, e_2\}) - MD(exam(Joe, pass), \{e_3\})$

  $0.42 + 0.48 * 0.58 - MD(exam(Joe, pass), \{e_3\}$

  MB(h,e) = $I(P_1 \wedge \ldots \wedge P_n \rightarrow_i h) * \max(0, \min(I(P_1), \ldots, I(P_n))$

  - $e_3 = work(X, high) \wedge extra(X, high) \rightarrow_3 \neg exam(X, pass)$ $\quad I(e_3) = 0.6$

  $MD(exam(Joe, pass), e_3) = 0.6 * \max(0, \min(0.4, 0.3)) = 0.18$

- if you know that
  - I(prep(Joe,good)) = 0.7
  - I(sleep(Joe, well)) = 0.6
  - I(contr(Joe,high)) = 0.9
  - I(work(Joe,low)) = 0.6
  - I(extra(Joe,high)) = 0.3

Examples (II)

- Interpret the statement

  Joe passes the exam

  I(exam(Joe,pass)) = ?

  $MB(exam(Joe, pass), \{e_1 , e_2\}) - MD(exam(Joe, pass), \{e_3\})$

  $0.42 + 0.48 * 0.58 - 0.18$

  0.5184

- if you know that
  - I(prep(Joe,good)) = 0.7
  - I(sleep(Joe, well)) = 0.6
  - I(contr(Joe,high)) = 0.9
  - I(work(Joe,low)) = 0.6
  - I(extra(Joe,high)) = 0.3

General Discussion

UNIVERSITY OF CALGARY

General Discussion

- Production rule systems can be seen as special logics based on operational semantics that take away the search aspect of the logics in 3.1.
- When using production systems, dealing with the control therefore requires more than just application knowledge and makes defining the knowledge base difficult.
- Newer approach: learning of rules by providing input-output pairs for the intended behavior

Onward to … frame systems

Jonathan Hudson, Ph.D.
jwhudson@ucalgary.ca
https://cspages.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY