

Introduction to Artificial Intelligence

8. Expert Systems

Prof. Jan Šnajder
Assoc. Prof. Marko Čupić
Prof. Bojana Dalbelo Bašić

University of Zagreb
Faculty of Electrical Engineering and Computing

Academic Year 2021/2022



Creative Commons Attribution-NonCommercial-NoDerivs 3.0

v0.3

Outline

- 1 What are expert systems and why do we need them?
- 2 Brief history and current state of affairs
- 3 Expert systems' architecture
- 4 Expert systems' inference
- 5 CLIPS

Outline

- 1 What are expert systems and why do we need them?
- 2 Brief history and current state of affairs
- 3 Expert systems' architecture
- 4 Expert systems' inference
- 5 CLIPS

GPS vs. ES

- Early AI (1950s and 1960s) focused on developing sophisticated inference techniques (e.g., automated theorem proving in FOL)
- The goal of the early systems was **general problem solving (GPS)**
- Early AI systems:
 - ▶ did not rely on substantial amounts of task-specific knowledge
 - ▶ tried to achieve intelligence without necessarily modeling human reasoning patterns
 - ▶ heavily relied on **search**
- The failure to develop a GPS gave rise to a different approach: **knowledge based systems** (aka **expert systems**)

Human knowledge

- Knowledge-based systems seek explicitly to capture what people know and how they use it
- How can one most easily represent **human knowledge**?
- Majority of human knowledge (expert knowledge) can be represented using **if-then rules**

Rule 1

If the temperature of the patient is greater than 38°C, then medications that lower the temperature should be prescribed.

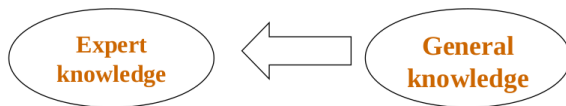
Rule 2

If the traffic light is red, then stop.

- These rules are also called **production rules**. The systems that use them are called **production systems**, **knowledge-based systems**, or **expert systems**

General vs. expert knowledge

- An important step in successful solving of AI problems: reducing the scope or domain of the problem



- **Expert knowledge** (domain knowledge) is knowledge **narrowly focused on a specific domain** (e.g., medicine, finances, chess etc.), unlike general problem solving knowledge
- No claim of breadth or generality!

Intellectual cloning



The overall intent is a form of **intellectual cloning**: find persons with a **reasoning skill that is important and rare** (e.g., an expert medical diagnostician, chess player, chemist), talk to them to determine what specialized knowledge they have and how they reason, then **embody that knowledge and reasoning in a program**.

(MITECTS, p430)

Expert systems vs. other approaches

• Human reasoning vs. theorem proving

- ▶ ES do not think much faster nor in fundamentally different ways than ordinary humans
- ▶ Their expertise arises because they have **substantially more knowledge about the task** than ordinary humans

• Knowledge vs. search

- ▶ Early systems tried to sidestep the need for knowledge with efficient search (e.g., chess playing)
- ▶ ES offered some of the earliest evidence that **knowledge could obviate the need for search**
- ▶ Combining search and knowledge: **knowledge-guided search** (e.g., DENDRAL)



Knowledge vs. search: DENDRAL

- DENDRAL (“Dendritic Algorithm”): **analytic chemistry** ES developed at Stanford University in 1960s, for determining the structure of a chemical compound from its mass spectrum
- **Generate-and-test**: generate possible structures and test them to see whether they would produce the mass spectrum observed. But this quickly led to **combinatorial explosion**
- By working with the **expert chemists**, DENDRAL’s authors were able to determine what clues chemists found in the spectra that permitted them to **focus search** on particular subclasses of molecules



Outline

- 1 What are expert systems and why do we need them?
- 2 Brief history and current state of affairs**
- 3 Expert systems' architecture
- 4 Expert systems' inference
- 5 CLIPS

Foundational ideas

① Production systems

- ▶ Logician Emil Post in 1943 proposed **production rules** in formal theoretical computer science
- ▶ a string-manipulation system that starts with finitely-many strings and repeatedly transforms them by applying a finite set of specified rules
- ▶ **Post canonical system** is reducible to a **string rewriting system (SRS)**. Both formalism are **Turing complete**

② Human cognition process

- ▶ 1972 book “Human problem solving” by Allen Newell and Herbert A. Simon: human cognition can be modeled by **if-then rules**
- ▶ a single rule is a single **chunk of knowledge**
- ▶ senses stimulate the brain with stimuli, which activate knowledge in our **long-term memory** \Rightarrow if-then rules
- ▶ **short-term memory** stores temporary knowledge produced when inference procedure is executed (chunks of knowledge that can be considered simultaneously) \Rightarrow new facts derived by rules

Early expert systems

- DENDRAL (Feigenbaum et al. 1971)
 - ▶ discovering molecular structure of substances based on spectrometry
- MYCIN (Davis 1977)
 - ▶ diagnosing and recommending treatment of bacterial blood infections
 - ▶ about 450 relatively independent if-then rules, **backward chaining**
 - ▶ the first system to have all of the hallmarks that came to be associated with expert systems
- XCON/R1 (McDermott 1978)
 - ▶ a system for configuring DEC VAX and PDP-11 computers
 - ▶ about 10,000 rules, **forward chaining**
- PROSPECTOR (Duda 1979)
 - ▶ estimating existence of ore based on geographical characteristics
 - ▶ uses **fuzzy logic** to model uncertain conclusions ⇒ TBD next lecture

ES: Success or failure?

- Early/mid 1980: first wave of commercial ES
 - Expensive and cumbersome to create (thousands of very expensive person hours for advanced systems)
 - Most abandoned between 1987 and 1992, but due to non-technical and non-economic issues (managerial and organizational issues, e.g., aligning the development with business strategy, legal issues, etc.)
- ⇒ Success of an ES in the technical or economic sense does not guarantee high levels of adoption or long-term use
- Still, opinions on the success of expert systems diverge. . .

ES: Success or failure?

In the 1990s and beyond, the term expert system and the idea of a standalone AI system mostly dropped from the IT lexicon. There are two interpretations of this. One is that **expert systems failed**: the IT world moved on because expert systems did not deliver on their over hyped promise. The other is the mirror opposite, that expert systems were simply **victims of their success**: as IT professionals grasped concepts such as rule engines, such tools migrated from being standalone tools for developing special purpose expert systems, to being one of many standard tools.

(Wikipedia: Expert system)

Once touted as potentially revolutionizing business operations, you don't hear much about expert systems these days, although proponents claim **more enterprises than you might expect have adopted them**.

(D. Haskin, "Years After Hype, 'Expert Systems' Paying Off For Some")

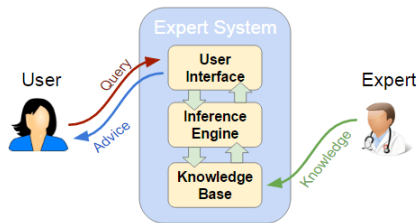
Outline

- 1 What are expert systems and why do we need them?
- 2 Brief history and current state of affairs
- 3 Expert systems' architecture**
- 4 Expert systems' inference
- 5 CLIPS

Inference engine vs. knowledge base

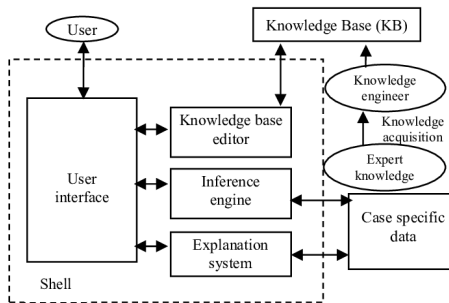
ES are built with different representation technologies, but all have two common architectural characteristics:

- 1 Distinction between **inference engine** and **knowledge base**
 - ▶ the inference engine retrieves rules from the knowledge base
- 2 Use of **declarative style representations**
 - ▶ rules are data structures with their own semantics, rather than part of the code implementing the inference engine



Expert systems shell

- Inference engine is decoupled from the knowledge base \Rightarrow different knowledge bases as “plug-ins”
- **Expert system shell:** a tool for building expert systems
 - ▶ an inference engine
 - ▶ knowledge base editor
 - ▶ user interface and explanation module



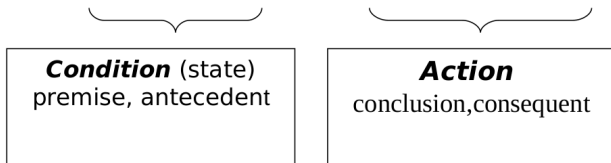
Popular expert systems shells

- CLIPS (“C Language Integrated Production System”), NASA (1985)
 - ▶ the most widely used ES shell, implemented in C
- JESS (“Java Expert System Shell”), Friedman-Hill & Ernest (2003)
 - ▶ CLIPS port to Java
- PyCLIPS
 - ▶ Python interface to CLIPS
- PyKE (“Python Knowledge Engine”)
 - ▶ Prolog-inspired ES shell implemented in Python
- ES-builder
 - ▶ educational ES, inference engine in Prolog

If-then rules

- Knowledge in ES is represented by if-then rules (production rules)

If **then**



- Reminiscent of **implication** in logic ($A \rightarrow B$), but two key differences:
 - implication is a formula of logic and has a truth value
 - the consequent B of an implication is also a formula, whereas the consequent of an if-then rule is an **action** (asserting new facts, but also deleting facts, executing code, e.g., printing on screen, etc.)

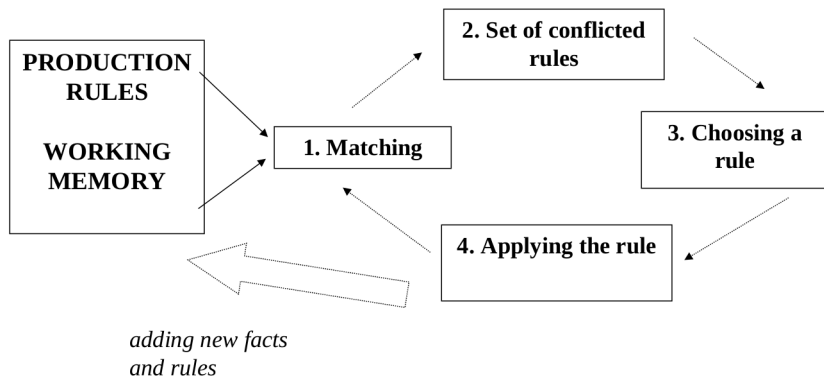
Outline

- 1 What are expert systems and why do we need them?
- 2 Brief history and current state of affairs
- 3 Expert systems' architecture
- 4 Expert systems' inference**
- 5 CLIPS

Inference components

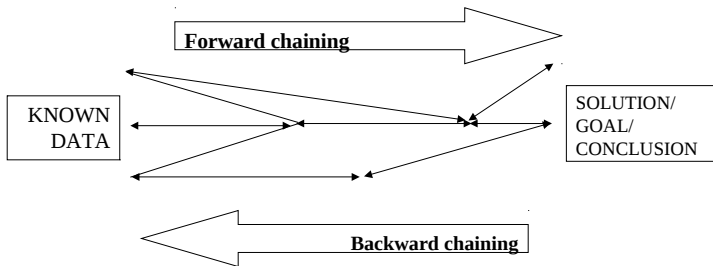
- **Working memory** is a part of knowledge base that:
 - ▶ stores facts added by the user before running inference or new facts derived during inference
 - ▶ does not store them permanently (akin to short-term memory in humans)
- **Inference engine** is a control mechanism carrying out the following:
 - ▶ **matching** – matching the facts from the working memory with the left-hand side (condition) of if-then rules (uses efficient **pattern-matching techniques**, e.g., the **Rete algorithm**)
 - ▶ **conflict resolution** – if more than one rule is enabled, selects one of them, e.g., the rule with the highest predefined priority (**“salience”**)
 - ▶ **rule application** (**“rule firing”**) – executing the right-hand side (action) of an if-then rule, which results in addition of new facts (**“assertion”**) to the working memory, deletion of facts (**“retraction”**) from working memory, or in any other action (rules may be also added to the knowledge base)

Inference cycle



14. INFERENCE PROCESS IN RULE-BASED SYSTEMS

- A sequence made of several conclusions that link the starting description of the problem with the solution is called a **CHAIN**



- Inference procedure, i.e. automatic advancement along the chain is called **chaining**.

14. INFERENCE PROCESS IN RULE-BASED SYSTEMS

There are two main ways of advancing towards conclusions:

1. FORWARD CHAINING

- starting with known data and advancing toward a conclusion (also called **forchaining**, **data driven processing**, event driven, bottom-up, antecedent, pattern directed **processing** ↔ **reasoning**)

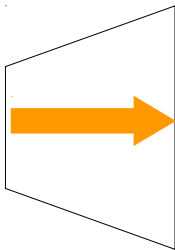
1. BACKWARD CHAINING

- choosing a possible conclusion (hypothesis) and trying to prove that it is valid by finding valid evidence.
(also called **backchaining**, **goal driven processing**, goal driven, top-down, consequent, expectation driven processing)

14. INFERENCE PROCESS IN RULE-BASED SYSTEMS

- **Forward chaining** - when there is a small amount of data and a large number of possible solutions
(for problem domains including synthesis :
for designing, planning, scheduling, supervision and
diagnostics of real time systems)

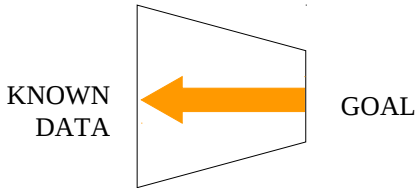
KNOWN
DATA



GOAL

14. INFERENCE PROCESS IN RULE-BASED SYSTEMS

- **Backward chaining** a reasonable choice when there aren't many possible conclusions and the amount of data known is large (Probleme diagnosing, classification)



- The choice of inference method depends on the properties of the problem domain and the way the expert reasons
- It is also possible to implement **bidirectional** inference.

15. FORWARD CHAINING

Example

- Rule base for determining type of fruit
- Parametres (i.e., variables) and their values are:

Shape:	elongated	Diameter:	> 10 cm
	circular		< 10 cm
	rounded	Fruit_type:	vine
Surface:	smooth		tree
	coarse	Fruit:	banana
Color:	green		watermelon
	yellow		melon
	brown-yellow		cantaloupe
	red		apple
	blue		apricot
	orange		cherry
Seed_number	> 1		peach
	= 1		plum
Seed_type	multiple		orange



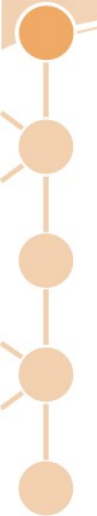
15. FORWARD CHAINING

RULES

- | | | | |
|---|------|---|----------------------|
| 1 | IF | Shape = elongated & Color = green or yellow | |
| | THEN | | Fruit = banana |
| 2 | IF | Shape= circular or rounded & Diameter > 10 cm | |
| | THEN | | Fruit_type = vine |
| 3 | IF | Shape= circular & Diameter < 10 cm | |
| | THEN | | Fruit_type = tree |
| 4 | IF | Seed_number = 1 | |
| | THEN | | Seed_type = bony |
| 5 | IF | Seed_number > 1 | |
| | THEN | | Seed_type = multiple |
| 6 | IF | Fruit_type = vine & Color = green | |
| | THEN | | Fruit = watermelon |



15. FORWARD CHAINING



7	IF	Fruit_type = vine & Surface = smooth & Color= yellow	
	THEN		Fruit = melon
8	IF	Fruit_type = vine & Surface = coarse & Color= brown-yellow	
	THEN		Fruit = cantaloupe
9	IF	Fruit_type = tree & Color= orange & Seed_type = bony	
	THEN		Fruit = apricot
10	IF	Fruit_type = tree & Color= orange & Seed_type = multiple	
	THEN		Fruit = orange
11	IF	Fruit_type = tree & Color = red & Seed_type = bony	
	THEN		Fruit = cherry
12	IF	Fruit_type = tree & Color = orange & Seed_type = bony	
	THEN		Fruit = peach
13	IF	Fruit_type = tree & Color = yellow or green or red & Seed_type = multiple	
	THEN		Fruit = apple
14	IF	Fruit_type = tree & Color = blue & Seed_type = bony	
	THEN		Fruit = plum



15. FORWARD CHAINING

- Known data: Diameter = 2 cm
Shape = circular
Seed_number = 1
Color = red

Conflict resolution strategy: the rule that is listed first

	<i>Working memory</i>	<i>Set of conflicting rules</i>	<i>Rule that fires</i>
0	Diameter = 2 cm Shape = circular Seed_number = 1 Color = red	3,4	3
1	Fruit_type: tree	3, 4	4
2	Seed_type = bony	3, 4, 11	11
3	Fruit = cherry	3, 4, 11	STOP

18. BACKWARD CHAINING

Backward chaining

- fundamentally different from forward chaining, even though both inference methods match and apply rules
- starts with a desired goal (hypothesis) and determines if the existing facts support proving the goal
- The system starts with an empty list of facts. A list of goals is given for which the system tries to determine the values.

18. BACKWARD CHAINING

1. Steps:

1. Form a stack containing the goals (hypotheses) that we want to prove.
2. On top of the stack is the goal which needs to be proven.
If the stack is empty STOP
3. Find **all** rules which can derive the current goal (find rules which have the desired goal on their **right hand side**)

18. BACKWARD CHAINING

4. For each of such rules do the following:

- 4.a. **If** all premises of the rule are satisfied (each premise parameter has the correct value in the working memory) **then** apply the rule, i.e. The RIGHT hand side of the rule i.e. add the conclusion to the working memory. Do not consider any more rules for that goal – its value has just been derived when the rule fired.
- If** the goal was a top goal **then** remove the goal from the stack & go back to step 2.
- If** the goal was an intermediate goal **then** remove the goal from the stack & return to the temporarily suspended goal

18. BACKWARD CHAINING

4.b. If the value of the parameter in working memory is not the same as the one in the premise

then do not apply that rule

4.c. If **premises of the rule are not satisfied** because one of the parameter values of the premise is not in working memory

then find **a rule** whose **right hand side derives the value of that parameter**

If at least one such rule exists

then set that parameter as an intermediate goal, i.e.
set that parameter on top of the stack &
go to step 2

18. BACKWARD CHAINING

4.d. If step(c) cannot find a rule which derives the required value of the parameter

then ask the user for the value of the parameter & add the value to working memory.

Go to step 4a and consider the next premise of the current rule

5. If all rules that could derive the current goal have been checked and if none of them could derive the goal

then goal remains underived. Remove the goal from the stack and go to step 2.

18. BACKWARD CHAINING

Example of backward chaining

- Let our starting hypothesis be that we are seeking a fruit.
- **hypothesis = (fruit).** Knowing that it is a cherry let's follow the backward chaining procedure to see if it is possible to derive that the fruit is a cherry.
- In the second step we consider ALL rules which derive the value of *fruit*

18. BACKWARD CHAINING

Goal (stack)	WORKING MEMORY	SET OF CONFLICTING RULES	Parameter which is being checked
(fruit)		1,6,7,8,9,10,11,12,13,14	shape is not in WM or on the RHS of a rule - ? shape?
(fruit)	shape circular = (R1)	6,7,8,9,10,11, 12,13,14	fruit_type - RHS of rules 2 i 3
(fruit_type Is the current goal; the goal fruit is temporarily suspended)		2,3	Shape - contained in WM - OK Diameter - not in WM or on the RHS of a rule ?diameter?
(fruit_type fruit)	diameter < 10 cm (R2)	3-FIRES, 6,7,8,9,10,11, 2,13,14	both premises of R3 are in WM
(fruit)	fruit_type = stablo	6,7,8,9,10,11, 12,13,14	first premise of R6 is not satisfied
(fruit)		(R6) 7,8,9,10,11, 12,13,14	first premise of R7 is not satisfied
(fruit)		(R7) 8,9,10,11, 12,13,14	first premise of R8 is not satisfied



18. BACKWARD CHAINING

(fruit)		(R8)	9,10,11, 12,13,14	fruit_type is in WM; 2nd premise is not in WM and not at RHS of any of the rules ? color ?
(fruit)	color = red	(R9)	10,11, 12,13,14	Second premise of R10 is not satisfied
(fruit)		(R10)	11, 12,13,14	
....
(fruit)			11 - FIRES	fruit = cherry

? ... ? – query to the user. All rules removed from the top of the stack for which it is not specified that they fire were not satisfied. Chaining goes backward until until the right hand side of the rule appears on the left hand side of another rule.



Outline

- 1 What are expert systems and why do we need them?
- 2 Brief history and current state of affairs
- 3 Expert systems' architecture
- 4 Expert systems' inference
- 5 CLIPS



EKSPERTNI SUSTAVI

Kroz primjere

I Marin Japec

"I'm sorry Dave, I'm afraid I can't do that. *Hall 9000*

CLIPS

Sintaksa nalik LISP-u (okrugle zagrade kao delimiteri)

Razvijen 1984, u NASA centru

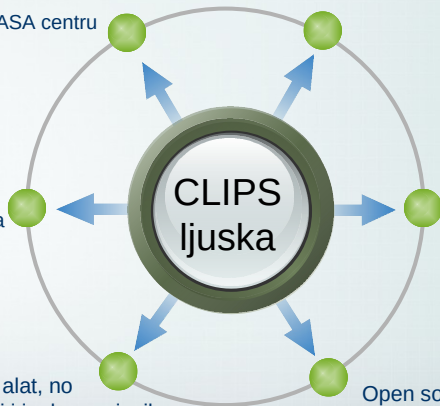
Potpuno okruženje za
razvoj ekspertnih sustava

Radi kao samostalan alat, no
moguće ga je pozvati i iz drugog jezika

Open source

Više paradigmatško
programsko okruženje:
(3 načina prikaza znanja)

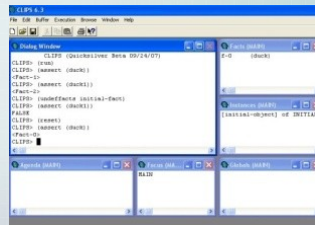
1. pravilima
2. proceduralno
3. objektno-orijentirano



- ❑ Razlikuje velika i mala slova
- ❑ Činjenice (*facts*) - Baza činjenica predstavlja početno stanje problema
- ❑ Pravila (*rules*) - Baza pravila sadrži operatore koji pretvaraju stanje problema u rješenje

- 1.Uspoređuje činjenice sa pravilima
- 2.Izabire koje pravilo izvršiti
- 3.Izvršava odgovarajuću akciju zadanu uz pravilo

“Watch”	<i>CLIPS> (watch rules)</i>
“Reset”	<i>CLIPS> (reset)</i>
“Run”	<i>CLIPS> (run)</i>



CLIPS-FACTS!

ČINJENICE

- ❑ Pregledavanje baze činjenica
- ❑ Dodavanje podatka u bazu činjenica: **"assert"**
- ❑ Brisanje činjenice: **"retract"**
- ❑ Brisanje svih činjenica: **"clear"**
- ❑ Definiranje više činjenica odjednom: **"deffacts"**
- ❑ Ili...učitavanje iz datoteke! Potrebno napraviti **"reset"** (tek tada dodajemo ih u bazu)
- ❑ Korištenje predložaka(*templates*): **"deftemplate"**

(*deftemplates Simpson*
(*slot ime (type STRING))*
(*slot godine (type NUMBER)*
(*default 36)*))



CLIPS> (*deffacts Simpsoni*
(*Simpson (ime Homer))*
(*Simpson (ime Marge) (godine (34)))*)



CLIPS> (*facts*)

CLIPS> (*assert (Homer voli pivo)*)

CLIPS> (*retract 0*)

CLIPS> (*clear*)

CLIPS> (*deffacts Simpsoni*
(*Homer voli pivo*)
(*Marge ima plavu kosu*))

CLIPS-RULES

PRAVILA

Sintaksa:

```
(defrule <imePravila>
<komentar(opcija)>
<deklaracija(opcija)>
<premisa1>
...
<premisaN>
=>
<akcija1>
...
<akcijaM>
)
```

Npr:

```
(defrule navike
"Homerove navike"
(salience 10)
(Homer drži pivo u ruci)
(Moe razgovara sa Homerom)
=>
(assert (Homer se nalazi u baru) )
(assert (Homer je sretan) )
)
```

Sa varijablama

```
(defrule navike
(?osoba drži pivo u ruci)
(Moe razgovara sa ?osoba))
=>
(assert (?osoba se nalazi u baru) )
(assert (?osoba je sretan) )
)
```



- Važnost pravila: "**salience**", raspon: [-10 000,10 000], veći broj, veća važnost, default 0

Vidi Clips User's Guide za više informacija!

Wrap-up

- Expert systems (knowledge-based systems) mimic **expert reasoning in a narrow domain**, rather than attempting general problem solving
- Knowledge represented with **if-then rules** (“chunks of knowledge”)
- A clear division between a **knowledge base** (stores rules and facts) and **inference engine** (does pattern matching, conflict resolution, and rule firing)
- An **expert system shell** provides an inference engine and a user interface that can be combined with different knowledge bases
- Inference may be implemented as **forward (data-driven) chaining** or **backward (goal-driven) chaining**, depending on use case
- **CLIPS** is a widely used expert system shell with forward chaining



Next topic: Modeling uncertainty