Metody Inżynierii Wiedzy

Listy decyzyjne. Tablice decyzyjne. Drzewa decyzyjne. Systemy regułowe. Systemy ekspertowe. Business Rules

Antoni Ligęza

Katedra Automatyki
Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki
Akademia Górniczo-Hutnicza
Kraków

2010

Wykaz literatury

1. Stuart J. Russel, Peter Norvig: *Artificial Intelligence. A Modern Approach.* Second Edition. Prentice Hall, New Jersey, 2003.
2. Jay Liebowitz: *The Handbook of Applied Expert Systems.* CRC Press, Boca Raton, 1998.
3. Frank van Harmelen, Vladimir Lifschitz, Bruce Porter (Eds.): *Handbook of Knowledge Representation*. Elsevier B.V., Amsterdam, 2008.
4. Michael Negnevitsky: *Artificial Intelligence. A Guide to Intelligent Systems.* Addison-Wesley, Pearson Education Limited, Harlow, England, 2002.
5. Adrian A. Hopgood: *Intelligent Systems for Engineers and Scientists.* CRC Press, Boca Raton, 2001.
6. Joseph C. Giarratano, Gary D. Riley: *Expert Systems. Principles and Programming.* Fourth Edition, Thomson Course Technology, 2005.

Wykaz literatury – c.d.

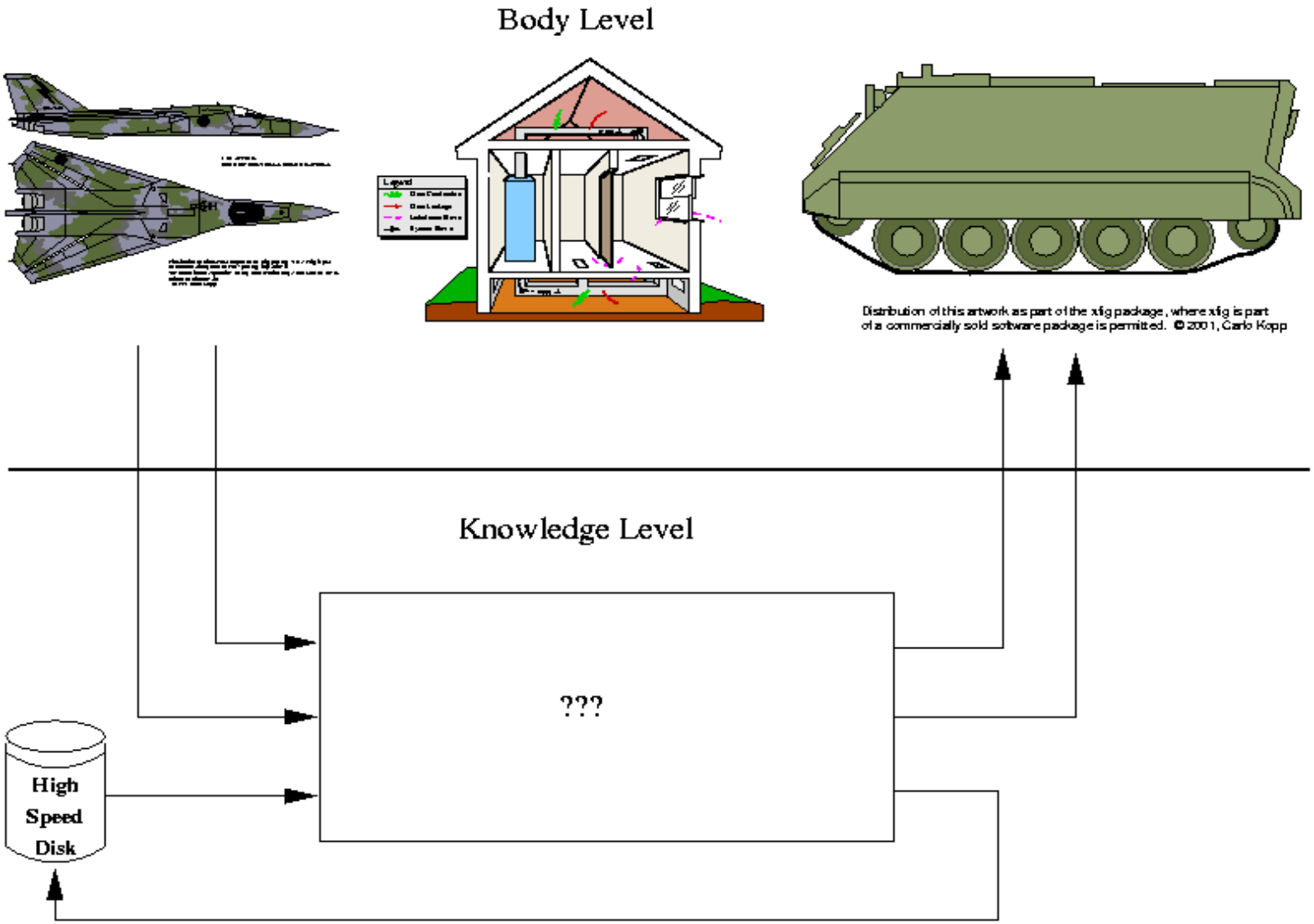
1. Peter Jackson: *Introduction to Expert Systems.* Addison-Wesley, Harlow, England, 1999.
2. Mordechai Ben-Ari: *Mathematical Logic for Computer Science.* Springer-Verlag, London, 2001.
3. Antoni Ligęza: *Logical Foundations for Rule-Based Systems.* Springer-Verlag, berlin, 2006.
4. Michael R. Genesereth, Nils J. Nilsson: *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., Los Altos, California, 1987.
5. Zbigniew Huzar: *Elementy logiki dla informatyków*. Oficyna Wyawnicza Politechniki Wrocławskiej, Wrocław, 2007.
6. Paweł Cichosz: *Systemy uczące się.* WNT, Warszawa, 2000.
7. Jan J. Mulawka: *Systemy ekspertowe.* WNT, Warszawa, 1996.

Przykłady systemów regułowych

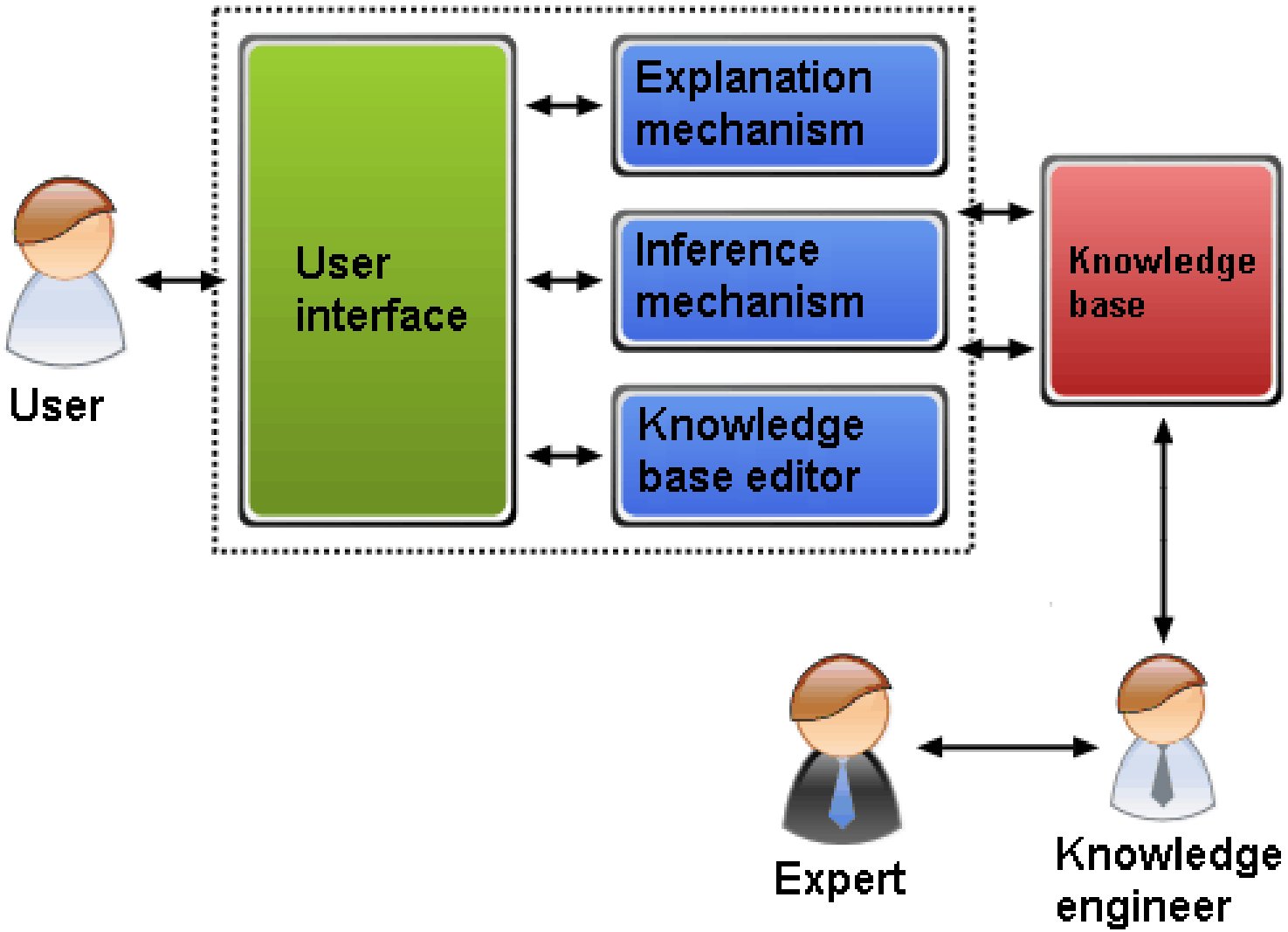
Obszary zastosowań

- społeczeństwo: reguły współżycia społecznego,
- prawo; nakazy i zakazy (KK, KPA, PoRD); podatki,
- organizacje (armia, firmy, banki, stowarzyszenia),
- szkolnictwo, szkolnictwo wyższe,
- ekonomia,
- socjologia, psychologia,
- medycyna,
- fizyka, chemia, matematyka, logika,
- biznes (BI – Business Intelligence, BR – Business Rules),
- technika:
  - sterowanie,
  - monitoring, kontrola ograniczeń, alarmy,
  - diagnostyka,
  - obsługiwanie, naprawa,
  - wspomaganie decyzji, systemy doradcze,
  - projektowanie i analiza, weryfikacja własności,
  - konfigurowanie.

Structure: Basic components of an Expert System Shell



Structure: Basic components of an Expert System Shell



Schemat bazy wiedzy systemu regułowego

Baza faktów

$$
\begin{aligned}
fact_1 &: \#q_1 \\
fact_2 &: \#q_2 \\
&\vdots \\
fact_k &: \#q_k
\end{aligned}
$$

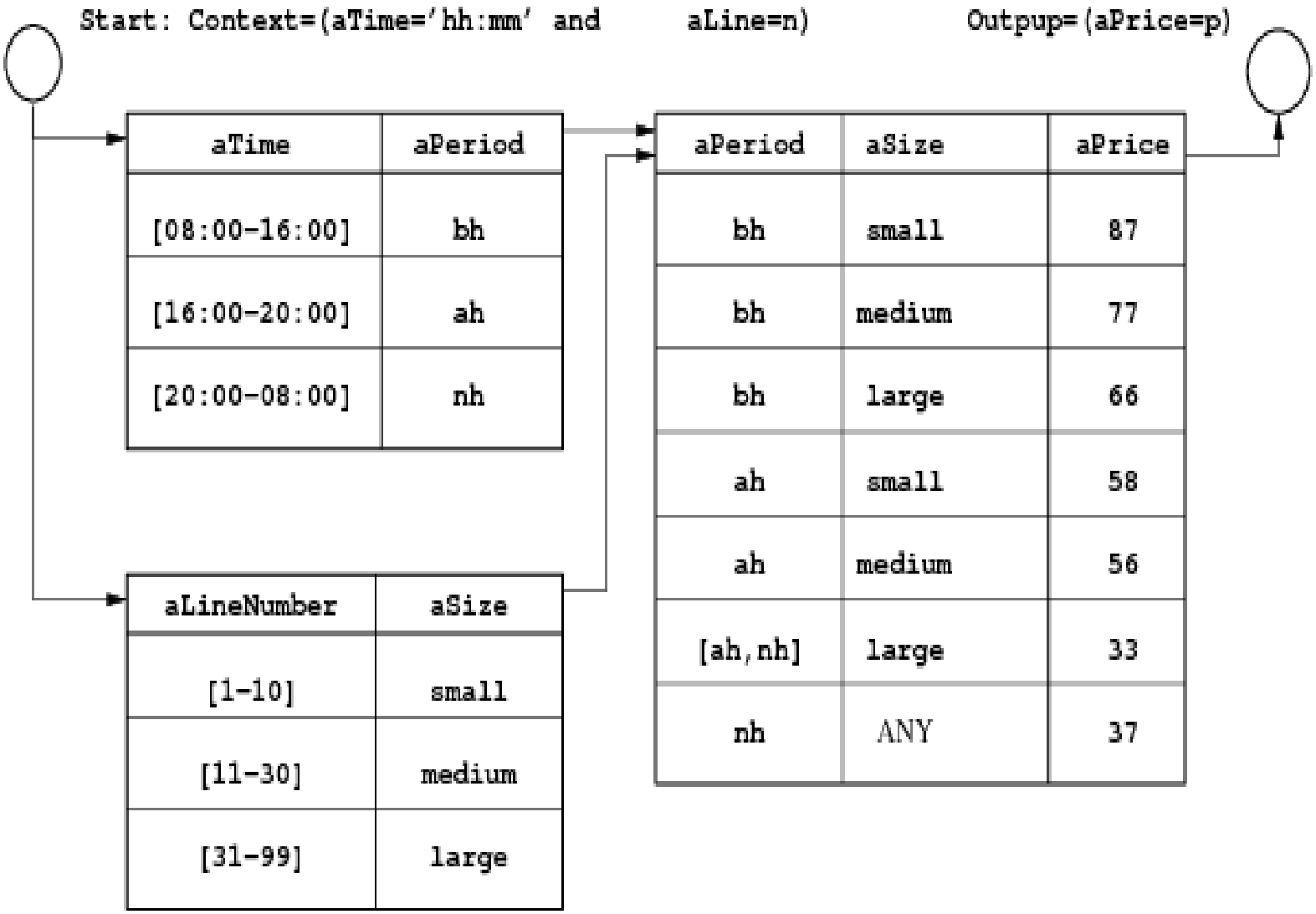
gdzie: $\#$ oznacza negację ($\neg$) lub jej brak; $\#p$ jest literałem.

Baza reguł

$$
\begin{aligned}
rule_1 &: \#p_1 \wedge \#p_2 \wedge \ldots \wedge \#p_n \longrightarrow \#h_1 \\
rule_2 &: \#p_1 \wedge \#p_2 \wedge \ldots \wedge \#p_n \longrightarrow \#h_2 \\
&\vdots \\
rule_m &: \#p_1 \wedge \#p_2 \wedge \ldots \wedge \#p_n \longrightarrow \#h_m
\end{aligned}
$$

gdzie: $\#$ oznacza negację ($\neg$) lub jej brak; $\#p$ jest literałem.

Example structure and rules



aTime	aPeriod
[08:00-16:00]	bh
[16:00-20:00]	ah
[20:00-08:00]	nh

aLineNumber	aSize
[1-10]	small
[11-30]	medium
[31-99]	large

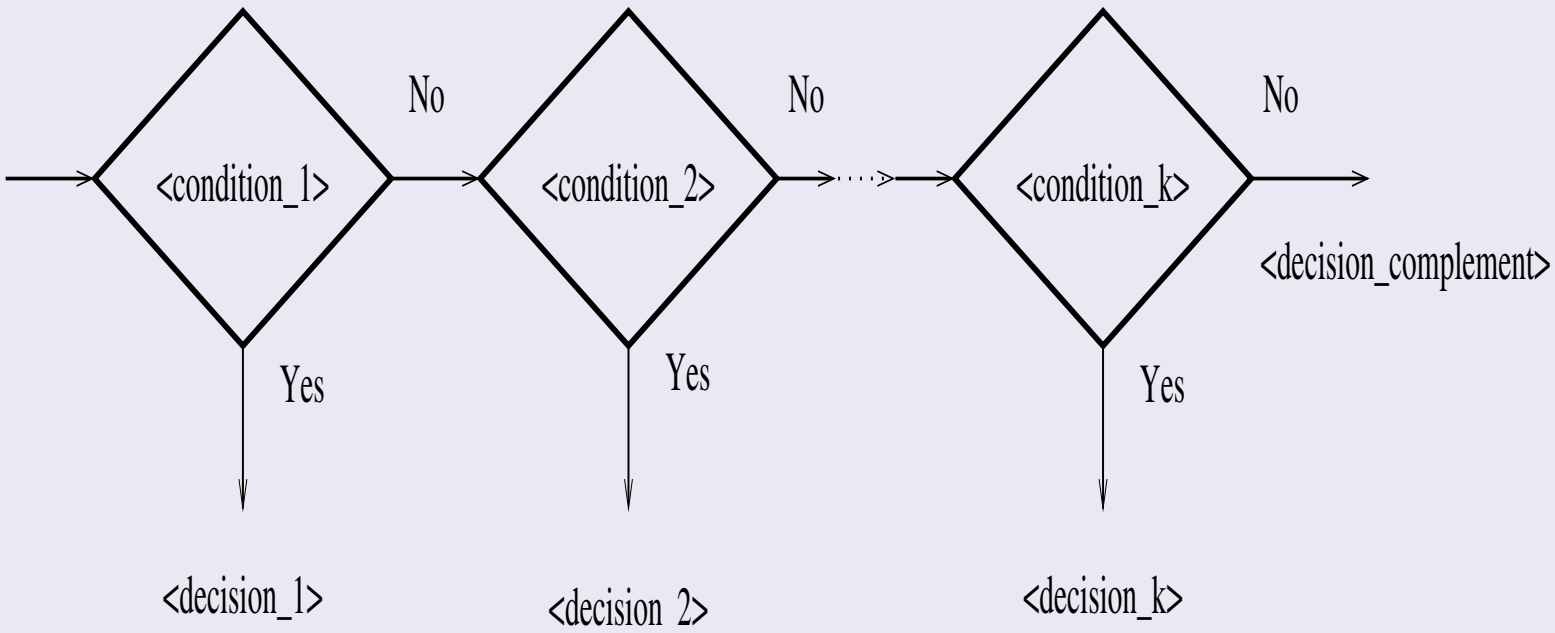
aPeriod	aSize	aPrice
bh	small	87
bh	medium	77
bh	large	66
ah	small	58
ah	medium	56
[ah,nh]	large	33
nh	ANY	37

Lista decyzyjna

Decision list scheme

```
if <condition_1> then <decision_1> else
if <condition_2> then <decision_2> else
...
if <condition_k> then <decision_k> else
   <decision_complement>
```

Decision list – picture



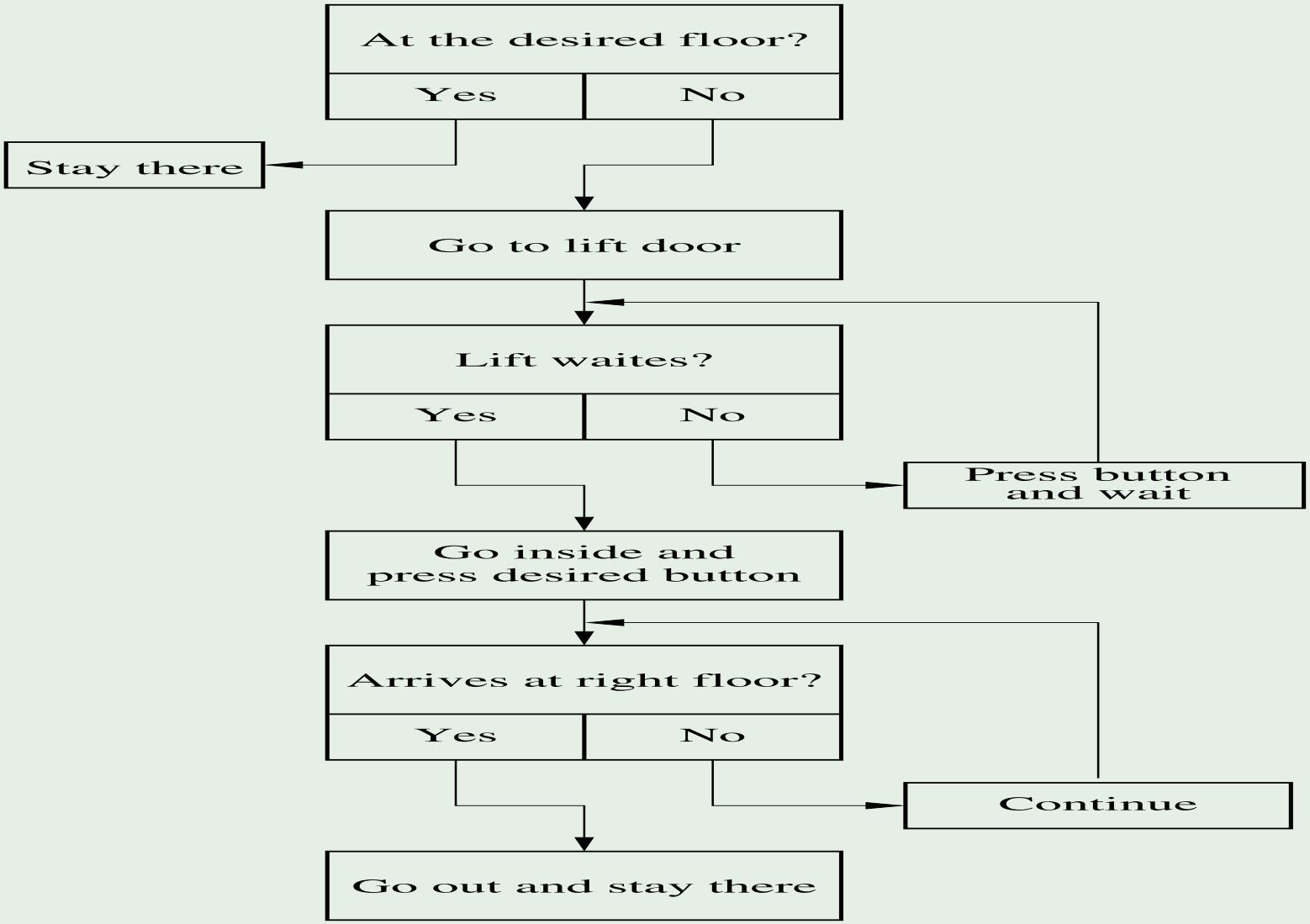
Lista decyzyjna – przkład z windą

- przykładem zachowania sterowanego regułami jest korzystanie z windy;
- winda jest systemem fizycznym sterowanym systemem regułowym;
- winda jest sterowana sterownikiem PLC.

Korzystanie z windy: algorytm

1. Check if you are at the desired floor – if so, stay there; if not – continue.
2. Go to the lift door (unconditional).
3. If the lift waits, go inside; if not, press the button and wait.
4. If you are inside, press the desired button and go.
5. If you arrive at certain floor, check if this is the desired floor and if so – go out; if not, continue your travel.

Korzystanie z windy: lista decyzyjna



Lista decyzyjna – przkład z windą

Korzystanie z windy: forma regułowa kontekstowa

$$
\begin{aligned}
\textit{Atthedesiredfloor}? &\longrightarrow \textit{Staythere} \\
\neg(\textit{Atthedesiredfloor}?) &\longrightarrow \textit{Gotoliftdoor} \\
\textit{Liftwaits}? &\longrightarrow \textit{Goinside} \\
&\quad \textit{andpressdesiredbutton} \qquad (1) \\
\neg(\textit{Liftwaits}?) &\longrightarrow \textit{Pressbuttonandwait} \\
\textit{Arrivesatrightfloor}? &\longrightarrow \textit{Gooutandstaythere} \\
\neg(\textit{Arrivesatrightfloor}?) &\longrightarrow \textit{Continue}
\end{aligned}
$$

Lista decyzyjna – przkład z windą

Korzystanie z windy: forma regułowa bezkontekstowa

$$
\begin{aligned}
\textit{Atthedesiredfloor}? &\longrightarrow \textit{Staythere} \\
\neg(\textit{Atthedesiredfloor}?) &\longrightarrow \textit{Gotoliftdoor} \\
\neg(\textit{Atthedesiredfloor}?) \wedge \textit{Liftwaits}? &\longrightarrow \textit{Goinside} \\
&\quad \textit{andpressdesiredbutton} \qquad (2) \\
\neg(\textit{Atthedesiredfloor}?) \wedge \neg(\textit{Liftwaits}?) &\longrightarrow \textit{Pressbuttonandwait} \\
\textit{Arrivesatrightfloor}? &\longrightarrow \textit{Gooutandstaythere} \\
\neg(\textit{Arrivesatrightfloor}?) &\longrightarrow \textit{Continue}
\end{aligned}
$$

Lista decyzyjna – przkład z windą

Korzystanie z windy: forma regułowa

$rule(1):$	$Atthedesiredfloor?$	$\longrightarrow$	$Staythere$		
			$next()$	$else(2)$	
$rule(2):$		$\longrightarrow$	$Gotoliftdoor$		
			$next(3)$	$else()$	
$rule(3):$	$Liftwaits?$	$\longrightarrow$	$Goinsideand$		
			$pressdesiredbutton$		
			$next(5)$	$else(4)$	(3)
$rule(4):$		$\longrightarrow$	$Pressbuttonandwait$		
			$next(3)$	$else()$	
$rule(5):$	$Arrivesatrightfloor?$	$\longrightarrow$	$Gooutandstaythere$		
			$next()$	$else(6)$	
$rule(6):$		$\longrightarrow$	$Continue$		
			$next(5)$	$else()$	

Drzewa w matematyce

Drzewo

Drzewo to dowolny graf *acykliczny* i *spójny*.
Drzewo ukorzenione to dowolny graf *acykliczny* i *spójny* posiadający jeden wyróżniony węzeł – korzeń drzewa.

- każde dwa wierzchołki drzewa są połączone,
- dwa dowolne wierzchołki drzewa łączy dokładnie jedna ścieżka,
- usunięcie dowolnej krawędzi powoduje niespójność grafu,
- dodanie dowolnej krawędzi pomiędzy istniejącymi wierzchołkami wprowadza cykl,
- korzeniem może zostać dowolny węzeł drzewa.

Zastosowania

- reprezentacja hierarchii,
- struktury danych (termy, listy, indeksy),
- kodowanie algorytmów (drzewa decyzyjne).

Binary Decision Tree

Drzewo binarne

A *Binary Decision Tree T* defined on N, E and P is an acyclic, directed graph satisfying the following properties:

1. There is exactly one distinguished node $n_0 \in N$ having no parent node (no entry); it is called the *root node* or the *root* of the tree; we shall write $n_0 = root(T)$,
2. Any other node has exactly one parent node,
3. Any node is assigned a single propositional symbol and it has either two child nodes or none:
   - if it has two child nodes, it is a *condition node* and there are two links to child nodes, one referring to the case if the propositional symbol takes *true* and the other one to the case it is *false*;
   - if it has no child nodes, it is a *decision node* or a *leaf node*, and the assigned to it propositional symbol indicates the decision.

Drzewo – definicja rekurencyjna

A binary decision tree is either:

- a single node $n \in N$ assigned unnegated or negated decision defined by propositional symbol $p \in P$,
- a graph formed from a node $n_i \in N$ assigned some propositional symbol $p \in P$, two subtrees, the left one and the right one, and two arcs labelled with *false* and *true* leading from the node to these subgraphs.

Drzewo – głębokość węzłów

The depth of a node in a binary decision tree is defined recursively as follows:

- $depth(n_0) = 0$,
- $depth(n) = depth(parent(n)) + 1$.

Drzewo binarne – jako term

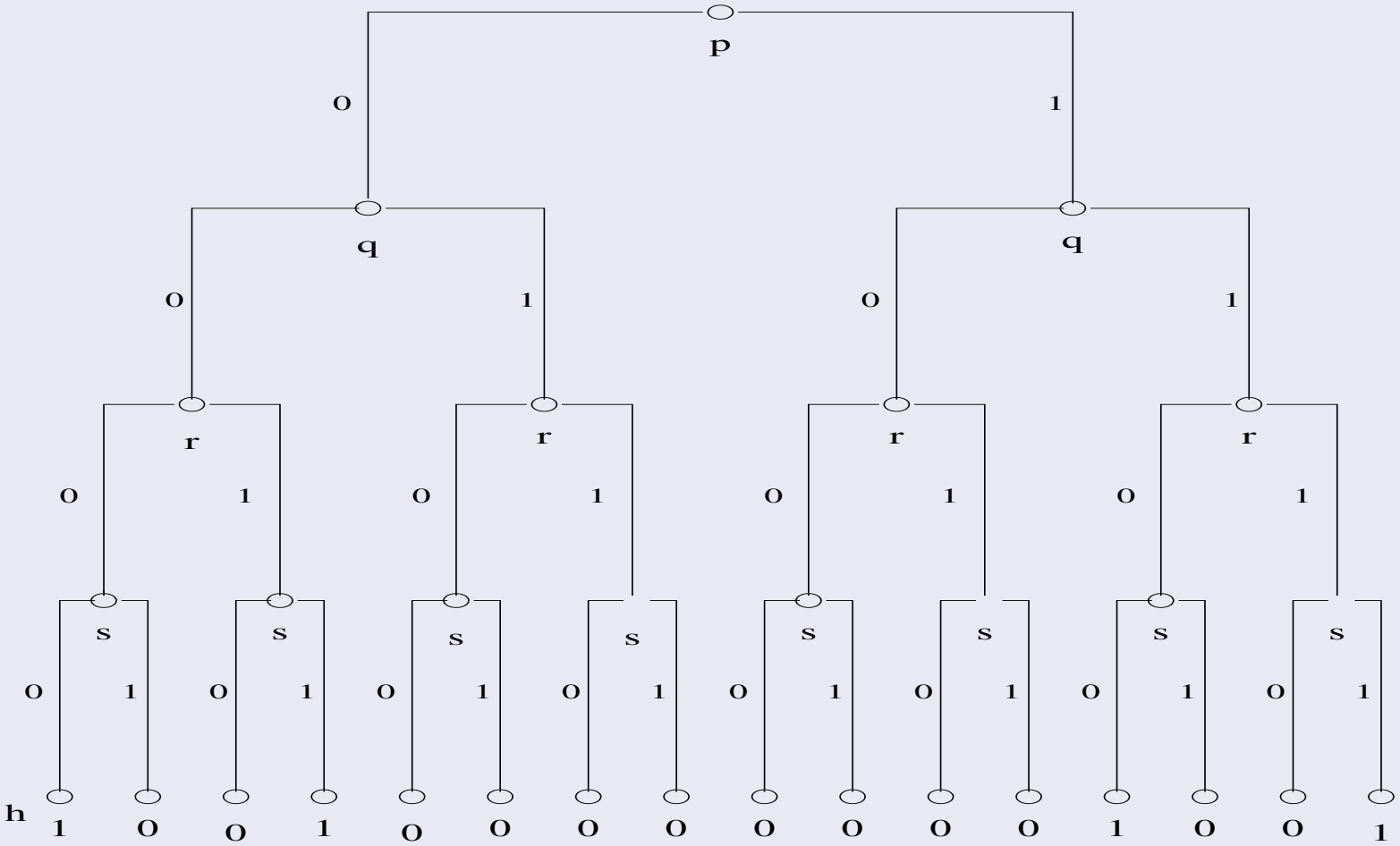
- $t(\_, \_, \#p)$ is a tree,
- if t_{left} and t_{right} are trees, then also $t(t_{left}, t_{right}, p)$ is a tree.

Ewaluacja formuły – metoda zero-jedynkowa

$$h \equiv (p \Leftrightarrow q) \wedge (r \Leftrightarrow s) \tag{4}$$

RuleNo	*p*	*q*	*r*	*s*	*h*
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

Drzewo binarne



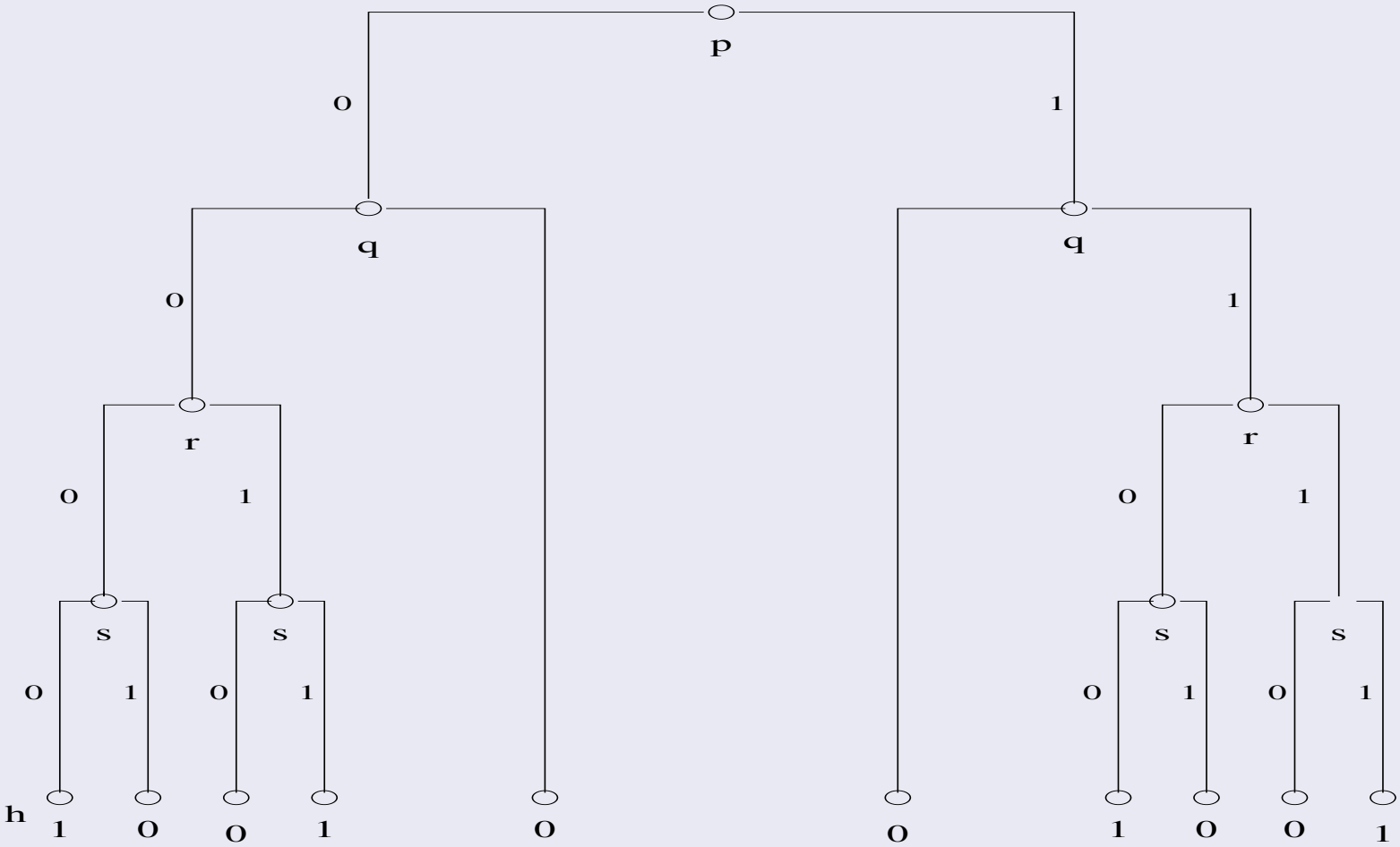
System regułowy

$$
\begin{aligned}
rule_0 &: \neg p \wedge \neg q \wedge \neg r \wedge \neg s \longrightarrow h \\
rule_3 &: \neg p \wedge \neg q \wedge r \wedge s \longrightarrow h \\
rule_{12} &: p \wedge q \wedge \neg r \wedge \neg s \longrightarrow h \\
rule_{15} &: p \wedge q \wedge r \wedge s \longrightarrow h
\end{aligned}
\tag{5}
$$

System regułowy – kompletacja

$$
\begin{aligned}
rule_1 &: \neg p \wedge \neg q \wedge \neg r \wedge s \longrightarrow \neg h \\
rule_2 &: \neg p \wedge \neg q \wedge r \wedge \neg s \longrightarrow \neg h \\
rule_4 &: \neg p \wedge q \wedge \neg r \wedge \neg s \longrightarrow \neg h \\
&\vdots \\
rule_{11} &: p \wedge \neg q \wedge r \wedge s \longrightarrow \neg h \\
rule_{13} &: p \wedge q \wedge \neg r \wedge s \longrightarrow \neg h \\
rule_{14} &: p \wedge q \wedge r \wedge \neg s \longrightarrow \neg h
\end{aligned}
\tag{6}
$$

Drzewo binarne zredukowane



Przegląd drzew

Gałęzienie: drzewa binarne i niebinarne; porządek lub brak,

Zawartość węzłów: warunki (testy kryteria), dane, struktury, decyzje; drzewo Trie - fragmenty kluczy,

Zastosowania: drzewa decyzyjne, klasyfikacja, indeksowanie,hierarchia, e

Efektywność dostępu: drzewa BST (Binary Search Trees); przechowują w węzłach klucze, lewe ($\leq$) i prawe poddrzewo ($\geq$),

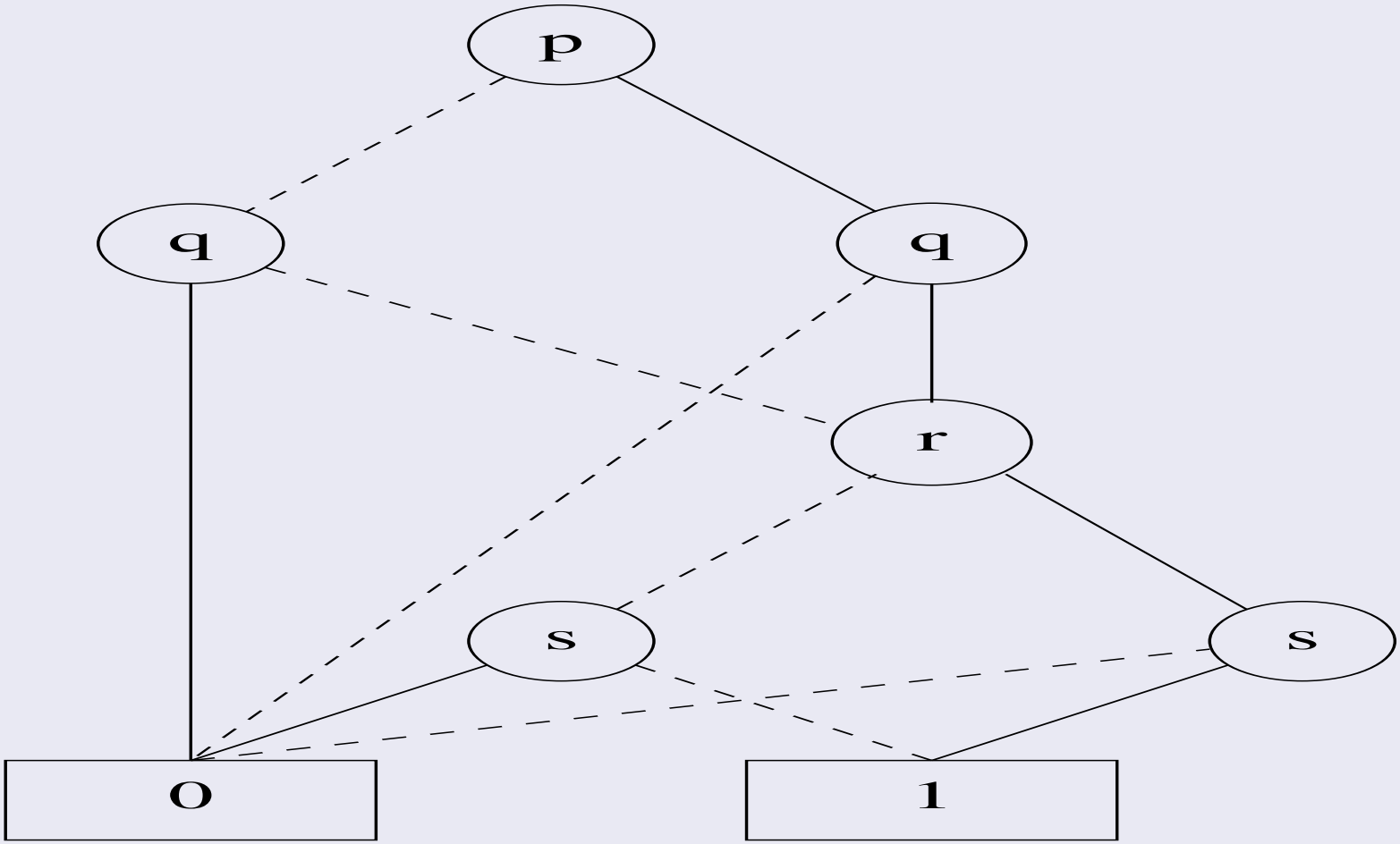
Efektywność dostępu: drzewa AVL () zrównoważone (ang. *balanced*) i niezrównoważone (ang. *unbalanced*); drzewa czerwono-czarne,

Efektywność dostępu: drzewa *trie* (retrieval); przechowuje w węzłach fragmenty kluczy.

Struktura: drzewa AST (Abstract Syntax Tree); termy, programy, funkcje,

Szukanie logiczne: drzewa binarne z warunkami logicznymi w węzłach.

Zredukowany OBDD



Tablice decyzyjne: schemat klasyczny (pionowy)

	rule_1	*rule_2*	...	*rule_m*
condition_1	v_{11}	v_{12}	...	v_{1m}
condition_2	v_{21}	v_{22}	...	v_{2m}
⋮	⋮	⋮		⋮
condition_n	v_{n1}	v_{n2}	...	v_{nm}
action_1	w_{11}	w_{12}	...	w_{1m}
action_2	w_{21}	w_{22}	...	w_{2m}
⋮	⋮	⋮		⋮
action_k	w_{k1}	w_{k2}	...	w_{km}

Tablice decyzyjne – notacja i ewaluacja

Notacja

- With respect to conditions:
  - $+$, T or Y if the condition must hold for certain action to be executed,
  - $-$, F or N if the condition cannot hold (if it holds, then the action cannot be executed),
  - _ if the execution of the action does not depend on the specific condition.

  With respect to conclusions:
  - X or $+$ if the action should be executed (or the conclusion should be drawn,
  - _ or $-$ if the action should be ignored.

Interpretacja

- any column of values specifying action prerequisites is traversed top-down, and the defined sequence of true and false conditions is verified;
- if the pattern is matched by the current state, the actions specified below are executed;
- next subsequent column of conditions is analyzed in a similar way.

Klasyczna tablica decyzyjna

	rule_1	*rule_2*	...	*rule_m*
p_1	v_{11}	v_{12}	...	v_{1m}
p_2	v_{21}	v_{22}	...	v_{2m}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
p_n	v_{n1}	v_{n2}	...	v_{nm}
h_1	w_{11}	w_{12}	...	w_{1m}
h_2	w_{21}	w_{22}	...	w_{2m}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
h_k	w_{k1}	w_{k2}	...	w_{km}

Example decision table

EU-Rent discounts for car rental

Car Group	Compact						Mid-size, Full-size												Lux, SUV, Van						Other	
Rental period	D, W, M				Other		D				W				M				Other		D		W		M		-	
New club member	Y		N		-		Y		N		Y		N		Y		N		-		-		-		-		-	
>3 days in advance	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
10%	x		x		x		x		x		x		x		x		x		x		x		x		x		x	
€ 50											x	x	x	x								x	x					
Max (10%, € 50)											x		x										x					
2-group upgrade	x	x					x	x			x	x			x	x												
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Decision tables are used by several commercial rule engines

Example from "Decision Tables and Business Rules", EBRC 2004 Tutorial, Jan Vanthienen

Atrybutowa tablica decyzyjna

att_1	*att_2*	...	*att_k*	*conc_1*	*conc_2*	...	*conc_m*
v_{11}	v_{12}	...	v_{1k}	w_{11}	w_{12}	...	w_{1m}
v_{21}	v_{22}	...	v_{2k}	w_{21}	w_{22}	...	w_{2n}
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
v_{n1}	v_{n2}	...	v_{nk}	w_{n1}	w_{n2}	...	w_{nm}

Tablica decyzyjna optyków

Oznacznia

- A_1 := age; $D_1 = \{y, p, q\}$, where:
  y – young,
  p – pre-presbyotic,
  q – presbyotic,
- A_2 := spectacle; $D_2 = \{m, h\}$, where:
  m – myope,
  h – hypermyope,
- A_3 := astigmatic; $D_3 = \{n, y\}$, where:
  n – no,
  y – yes,
- A_4 := tear production rate; $D_4 = \{r, n\}$, where: r – reduced,
  n – normal,
- D := type of contact lenses (decision attribute); $D_D = \{H, S, N\}$, where:
  H – Hard contact lenses,
  S – Soft contact lenses,
  N – No contact lenses.

Tablica decyzyjna optyków

Number	Age	Spectacle	Astigmatic	Tear p.r.	Decision
1	y	m	y	n	H
2	y	n	y	n	H
3	p	m	y	n	H
4	q	m	y	n	H
5	y	m	n	n	S
6	y	n	n	n	S
7	p	m	n	n	S
8	p	n	n	n	S
9	q	n	n	n	S
10	y	m	n	r	N
11	y	m	y	r	N
12	y	n	n	r	N
13	y	n	y	r	N
14	p	m	n	r	N
15	p	m	y	r	N
16	p	n	n	r	N
17	p	n	y	r	N
18	p	n	y	n	N
19	q	m	n	r	N
20	q	m	n	n	N
21	q	m	y	r	N
22	q	n	n	r	N
23	q	n	y	r	N
24	q	n	y	n	N

Rule-Based Systems: Basic Concepts

A rule

$$rule: \ < preconditions > \longrightarrow < conclusions >,$$

$< preconditions >$ — when the rule can be *fired*,
$< conclusions >$ — whatever follows from the rule application.

LHS and RHS of a rule

$$LHS(rule) =< preconditions >,$$

$$RHS(rule) =< conclusions/actions > .$$

A typical rule

$$p_1 \ AND \ p_2 \ AND \ \ldots \ AND \ p_n \longrightarrow h.$$

An example rule

$$preassure(normal) \wedge temperature(water, T) \wedge T > 100 \longrightarrow state(water, boiling)$$

Thermostat

Intelligent Temperature Control

- The goal of the system is to **set temperature** at a certain *set point*, which is the output of the system.
- The input is the current **time and date** (hour, day, date, month).
- The temperature is set depending on the particular part of the week, season, and working hours.

Note: the example is based on the book by M.Negnevitsky (2002).

Knowledge specification: Rules I

Rule 1 **if** the day is Monday **or** the day is Tuesday **or** the day is Wednesday **or** the day is Thursday **or** the day is Friday **then** today is a workday.

Rule 2 **if** the day is Saturday **or** the day is Sunday **then** today is the weekend.

Rule 3 **if** today is workday **and** the time is 'between 9 am and 5 pm' **then** operation is 'during business hours'.

Rule 4 **if** today is workday **and** the time is 'before 9 am' **then** operation is 'not during business hours'.

Rule 5 **if** today is workday **and** the time is 'after 5 pm' **then** operation is 'not during business hours'.

Rule 6 **if** today is weekend **then** operation is 'not during business hours'.

Rule 7 **if** the month is January **or** the month is February **or** the month is December **then** the season is summer.

Rule 8 **if** the month is March **or** the month is April **or** the month is May **then** the season is autumn.

Rule 9 **if** the month is June **or** the month is July **or** the month is August **then** the season is winter.

Knowledge specification: Rules II

Rule 10 **if** the month is September **or** the month is October **or** the month is November **then** the season is spring.

Rule 11 **if** the season is spring **and** operation is 'during business hours' **then** thermostat_setting is '20 degrees'.

Rule 12 **if** the season is spring **and** operation is 'not during business hours' **then** thermostat_setting is '15 degrees'.

Rule 13 **if** the season is summer **and** operation is 'during business hours' **then** thermostat_setting is '24 degrees'.

Rule 14 **if** the season is summer **and** operation is 'not during business hours' **then** thermostat_setting is '27 degrees'.

Rule 15 **if** the season is autumn **and** operation is 'during business hours' **then** thermostat_setting is '20 degrees'.

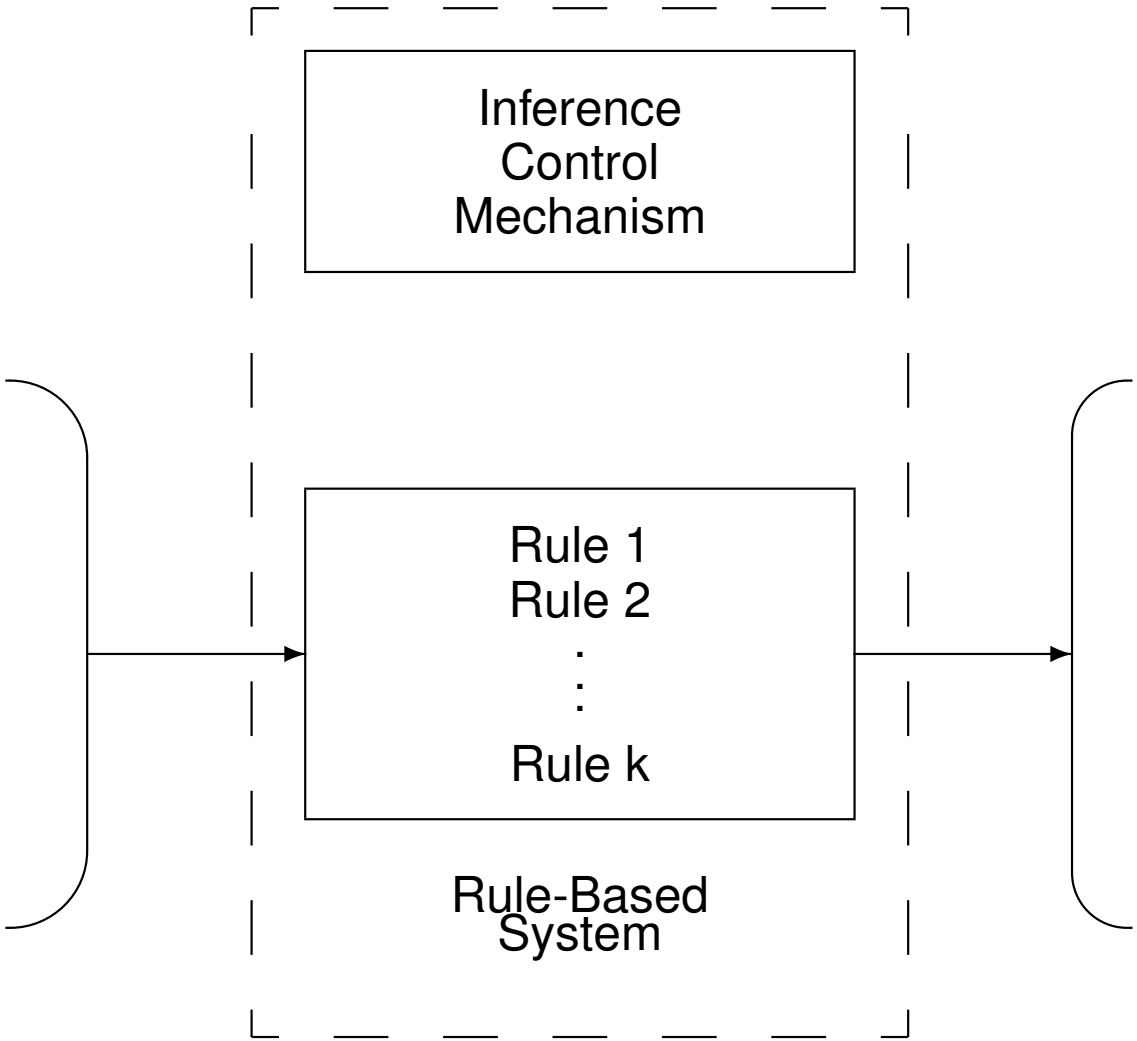
Rule 16 **if** the season is autumn **and** operation is 'not during business hours' **then** thermostat_setting is '16 degrees'.

Rule 17 **if** the season is winter **and** operation is 'during business hours' **then** thermostat_setting is '18 degrees'.

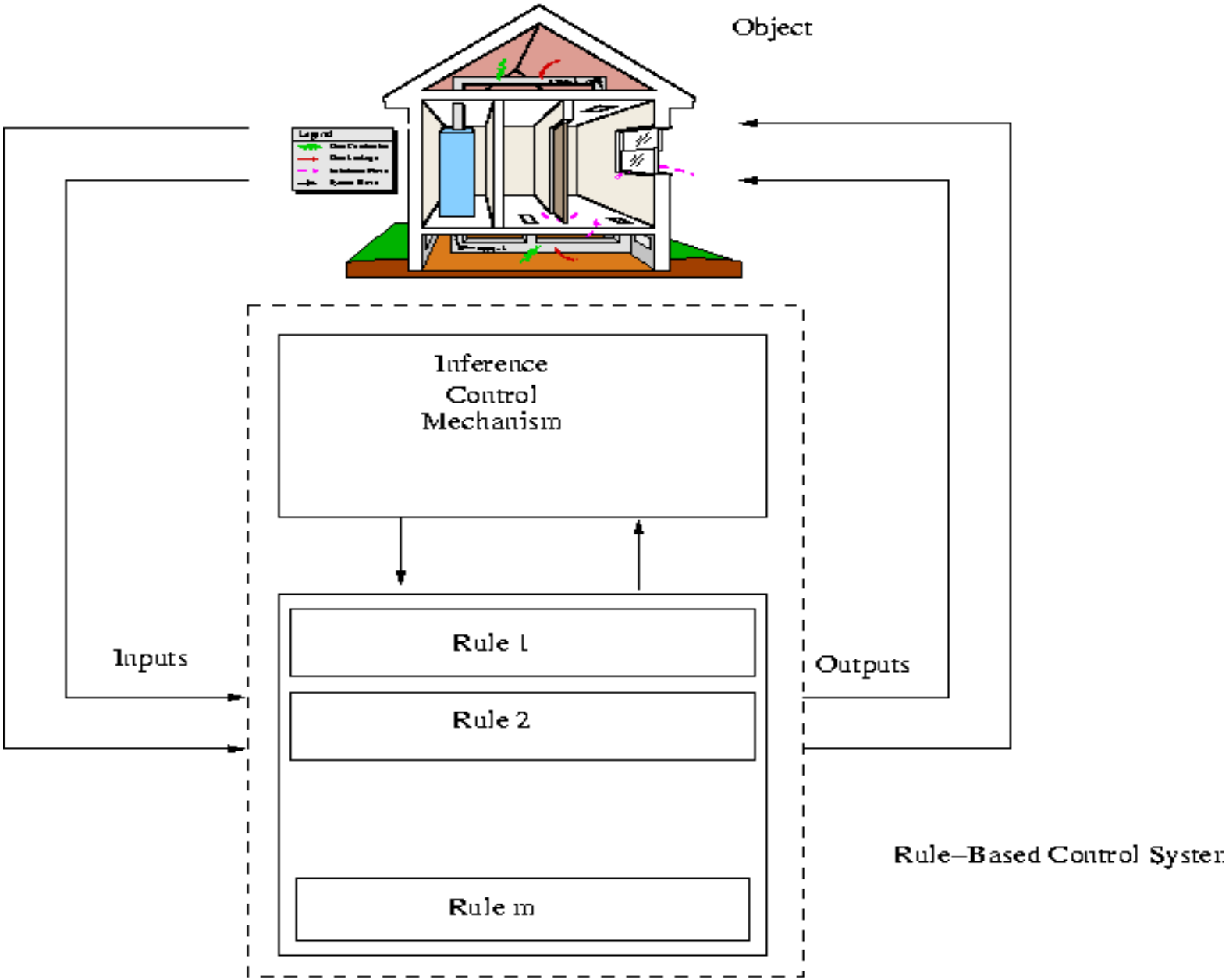
Rule 18 **if** the season is winter **and** operation is 'not during business hours' **then** thermostat_setting is '14 degrees'.

Note: There are certain patterns that may be observed in the rule-base. They will be used in the following section containing the corresponding XTT structure.

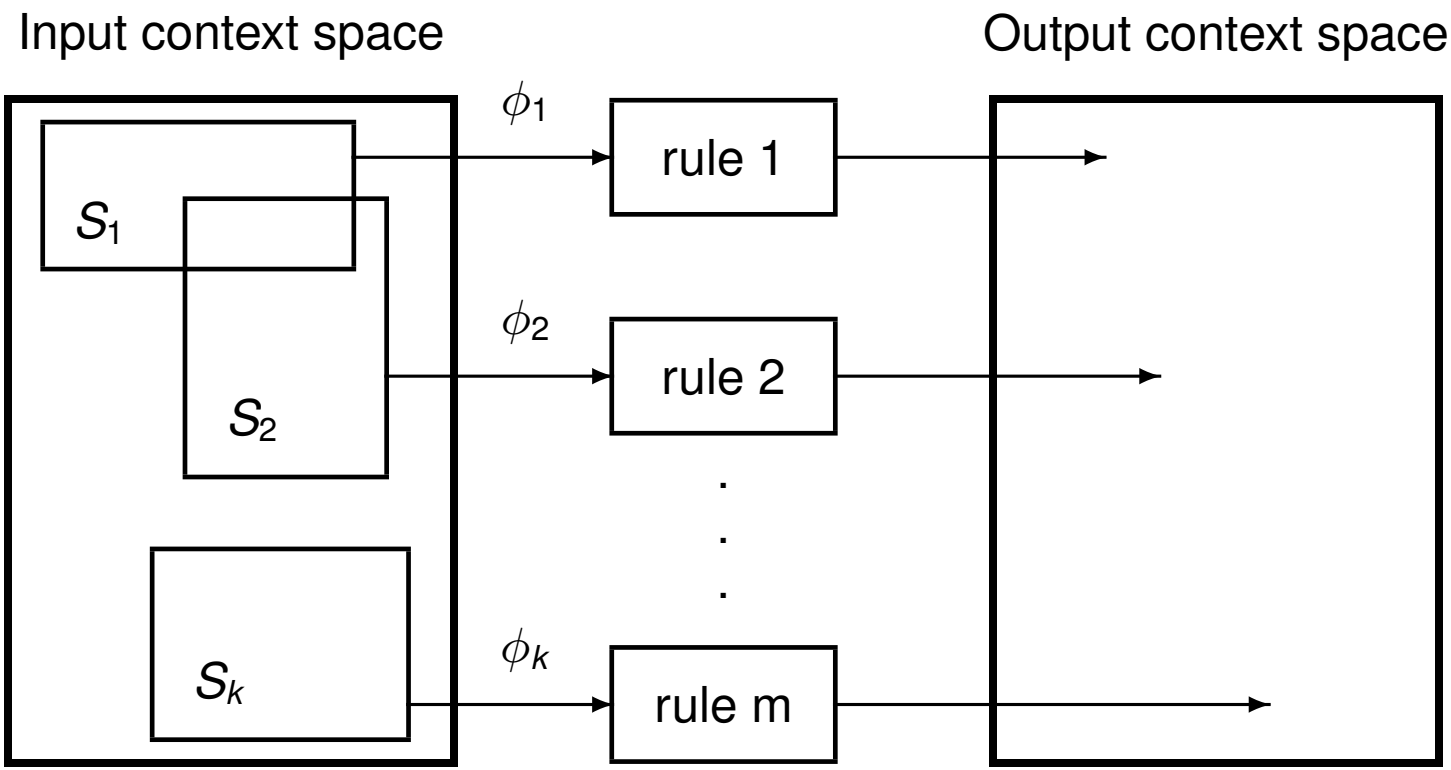
A general scheme of a rule-based system interaction with the environment



Structure: Basic components of an Expert System Shell



An abstract picture of the spatial interpretation



Reguły

Najprostsza reguła

$$r \longrightarrow p,$$

Reguła z koniunkcją pozytywnych prewarunków; Horn rule

$$rule: p_1 \wedge p_2 \wedge \ldots \wedge p_n \longrightarrow h.$$

Reguła z koniunkcją pozytywnych prewarunków i koniunkcją konkluzji

$$rule: p_1 \wedge p_2 \wedge \ldots \wedge p_n \longrightarrow h_1 \wedge h_2 \wedge \ldots h_k.$$

Dekompozycja reguły złożonej do reguł Horna

$$\begin{aligned} rule_1 &: p_1 \wedge p_2 \wedge \ldots \wedge p_n \longrightarrow h_1 \\ rule_2 &: p_1 \wedge p_2 \wedge \ldots \wedge p_n \longrightarrow h_2 \\ &\vdots \\ rule_k &: p_1 \wedge p_2 \wedge \ldots \wedge p_n \longrightarrow h_k \end{aligned}$$

Schemat systemu regułowego

Typowy schemat zbioru reguł

$$
\begin{aligned}
rule_1 &: \#p_1 \wedge \#p_2 \wedge \ldots \wedge \#p_n \longrightarrow \#h_1 \\
rule_2 &: \#p_1 \wedge \#p_2 \wedge \ldots \wedge \#p_n \longrightarrow \#h_2 \\
&\vdots \\
rule_m &: \#p_1 \wedge \#p_2 \wedge \ldots \wedge \#p_n \longrightarrow \#h_m
\end{aligned}
$$

Podział prewarunków na pozytywne i negatywne

$$LHS(rule) = LHS^{+}(rule) \wedge LHS^{-}(rule),$$

An example decomposition

$$p_1 \wedge \neg p_2 \wedge p_3 \wedge p_4 \wedge \neg p_5 \longrightarrow h$$

$$LHS^{+} \wedge LHS^{-} \longrightarrow h$$

$LHS^{+} = \{p_1, p_3, p_4\}$, $LHS^{-} = \{p_2, p_5\}$

Reguły o złożonych prewarunkach

Reguła o dowolnej formule prewarunków

$$rule: \Phi \longrightarrow h.$$

Dekompozycja prewarunków do postaci DNF

$$rule: \phi_1 \vee \phi_2 \vee \ldots \vee \phi_n \longrightarrow h.$$

Dekompozycja do postaci zbioru reguł o prewarunkach koninktywnych

$$\begin{aligned} rule_1 &: \phi_1 \longrightarrow h \\ rule_2 &: \phi_2 \longrightarrow h \\ &\vdots \\ rule_n &: \phi_n \longrightarrow h \end{aligned}$$

Aktywacja (odpalanie) reguł

Rule firing

Activation of rules depends on five main factors:

- the rule must be assigned the current context (situation),
- the preconditions of the rule are satisfied,
- the excluding conditions (if any) are unsatisfied,
- the resources (if any) must be available,
- the rule is selected by the inference engine.

Rule firing: further potential factors

- selection of rules useful for goal satisfaction (goal-oriented strategies),
- elimination of useless rules (rule-blocking strategies),
- combined selection of rules enabling goal-oriented rules,
- meta-rules: what rules should be used under what circumstances,
- rule-ordering strategies and conflict resolution.

General Problem Solver

Means-Ends Analysis: Match, find difference, reduce difference (apply rule) cycle.

Rule firing: logical conditions

State representation

$$\phi = \phi^+ \wedge \phi^-,$$

Satisfaction of preconditions

$$\phi \models LHS(rule)$$

$$\phi^+ \models LHS^+(rule),$$
$$\phi^- \models LHS^-(rule).$$

$$LHS^+(rule) \subseteq [\phi^+],$$
$$LHS^-(rule) \subseteq [\phi^-].$$

Excluding conditions

$$LHS^+(rule) \cap [\phi^-] \neq \emptyset,$$
$$LHS^-(rule) \cap [\phi^+] \neq \emptyset.$$

Monotonic versus non-monotonic systems

Assert and retract

- *retract(q)* – retracts (deletes) fact q from the knowledge base, and
- *assert(q)* – asserts (adds) fact q to the knowledge base.

A dynamic rule

$$rule: p_1 \wedge p_2 \wedge \ldots \wedge p_n \longrightarrow retract(d_1, d_2, \ldots, d_d), assert(h_1, h_2, \ldots, h_h).$$

RS trigger specification

$$Q' = Q \wedge \neg R \vee S.$$

$$\begin{aligned} rule_1: &\ Q \wedge R \wedge \neg S \longrightarrow retract(Q), assert(\neg Q) \\ rule_2: &\ \neg Q \wedge \neg R \wedge S \longrightarrow retract(\neg Q), assert(Q) \\ rule_3: &\ \neg R \wedge \neg S \longrightarrow do\_nothing(\neg Q) \end{aligned}$$

Inference modes

Basic inference modes w.r.t. direction

- **forwards** – forward-chaining, data-driven,
- **backwards** – backward-chaining, goal-oriented,
- **mixed** – combined strategies,
- **top-down** – top-down development of a partial order (plan).

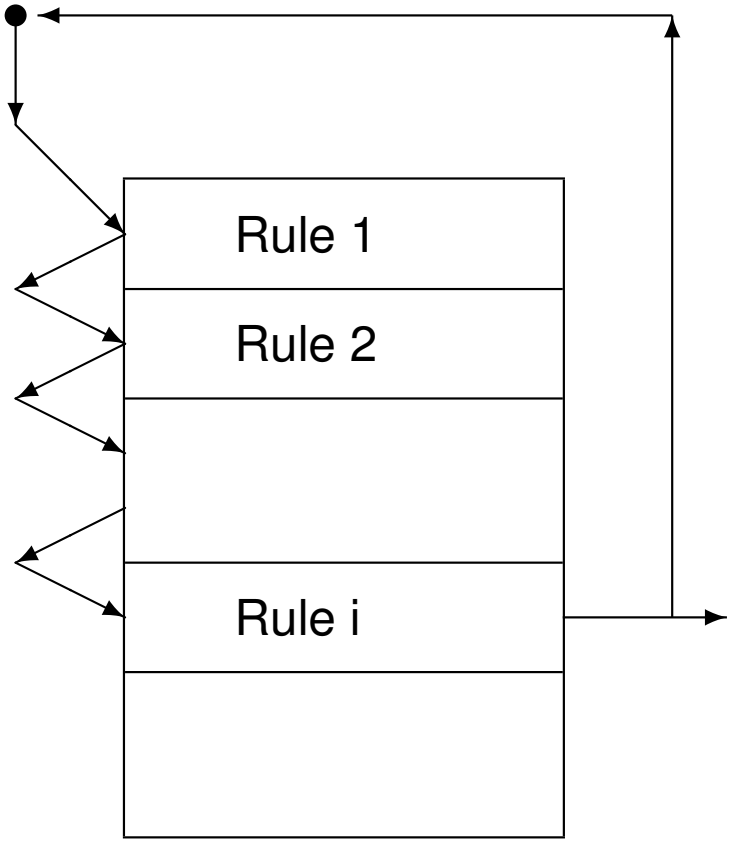
Basic inference modes w.r.t. depth

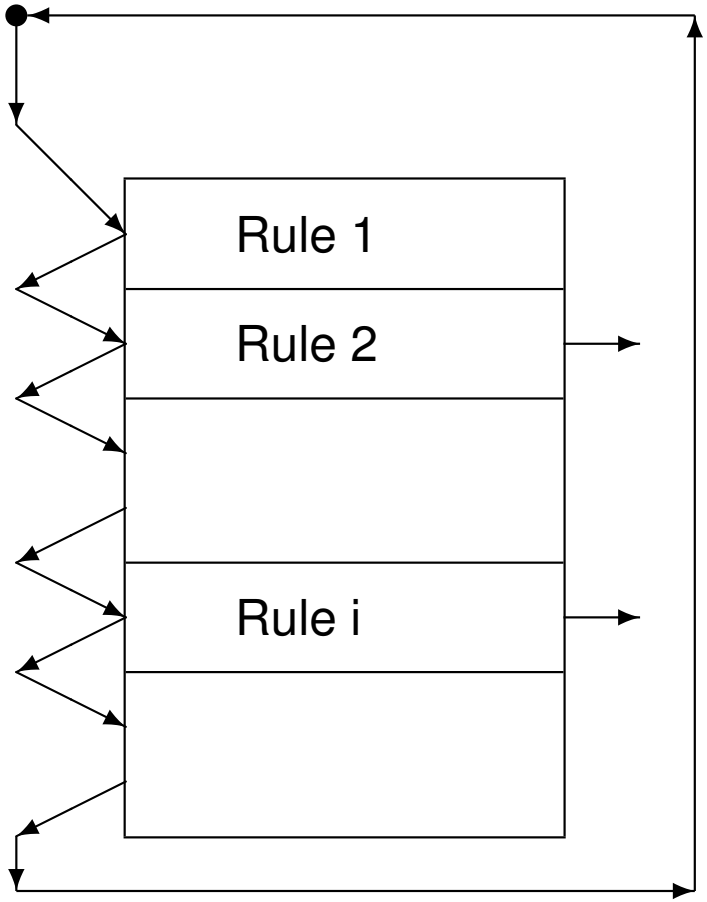
- **breadth-first** – level-saturation,
- **depth-first** – chained; limiting depth.
- **mixed**.

Basic forward-chaining strategies

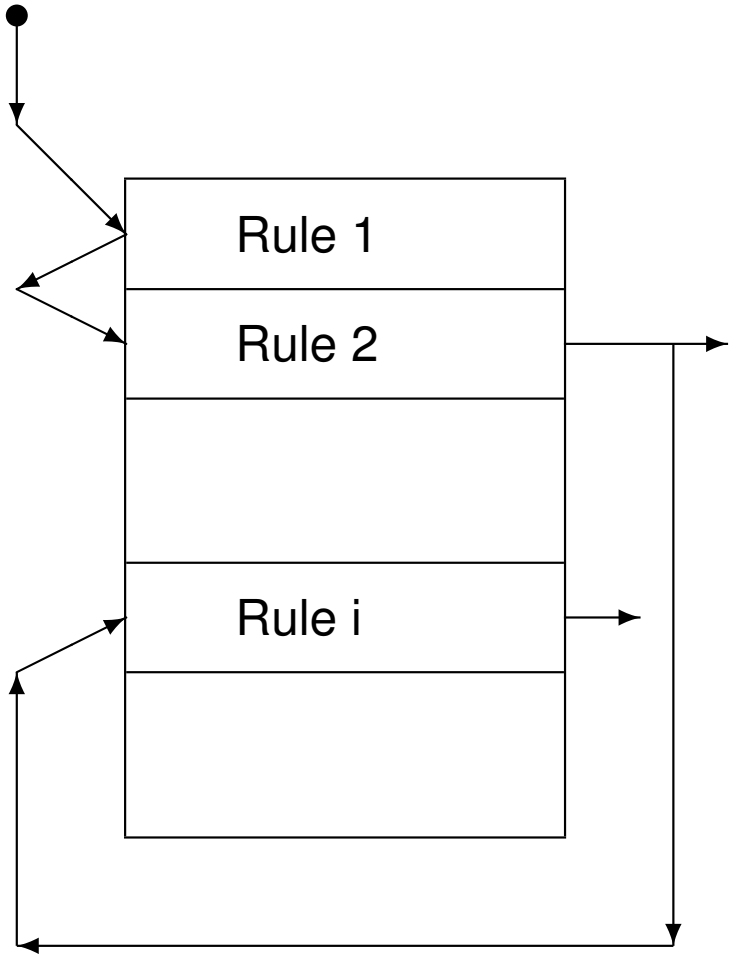
- **closed-loop hierarchical linear strategy** a hierarchy amongst the rules is assumed. The rules are tested sequentially until one with satisfied preconditions is found. Then the rule is executed and the process is resumed from rule 1. In PROLOG this would correspond to the *run-find-execute-!-run* loop scheme,
- **closed-loop linear strategy** the rules are ordered linearly and satisfiability of their preconditions is tested sequentially in a closed loop. After rule *i* rule number $i + 1$ is tested and possibly executed; after rule *n* rule 1 is taken into account. In PROLOG this scheme would correspond to the *repeat-find-execute-fail* loop format,
- **linear strategy with rule-switching** linear interpretation from the beginning to first executable rule (one with satisfied preconditions); then the system jumps to some rule or rules indicated by the recently executed rule. This kind of scheme consists in switching among rules. This approach corresponds to use of the *next* part in rules,
- **linear strategy with context-switching** finally, linear rule examination may be combined with context switching, i.e. after a rule is executed, the current situation of the system is evaluated, the current context (qualitative situation) is determined and a decision making mechanism switches the set of rules appropriate for this context.

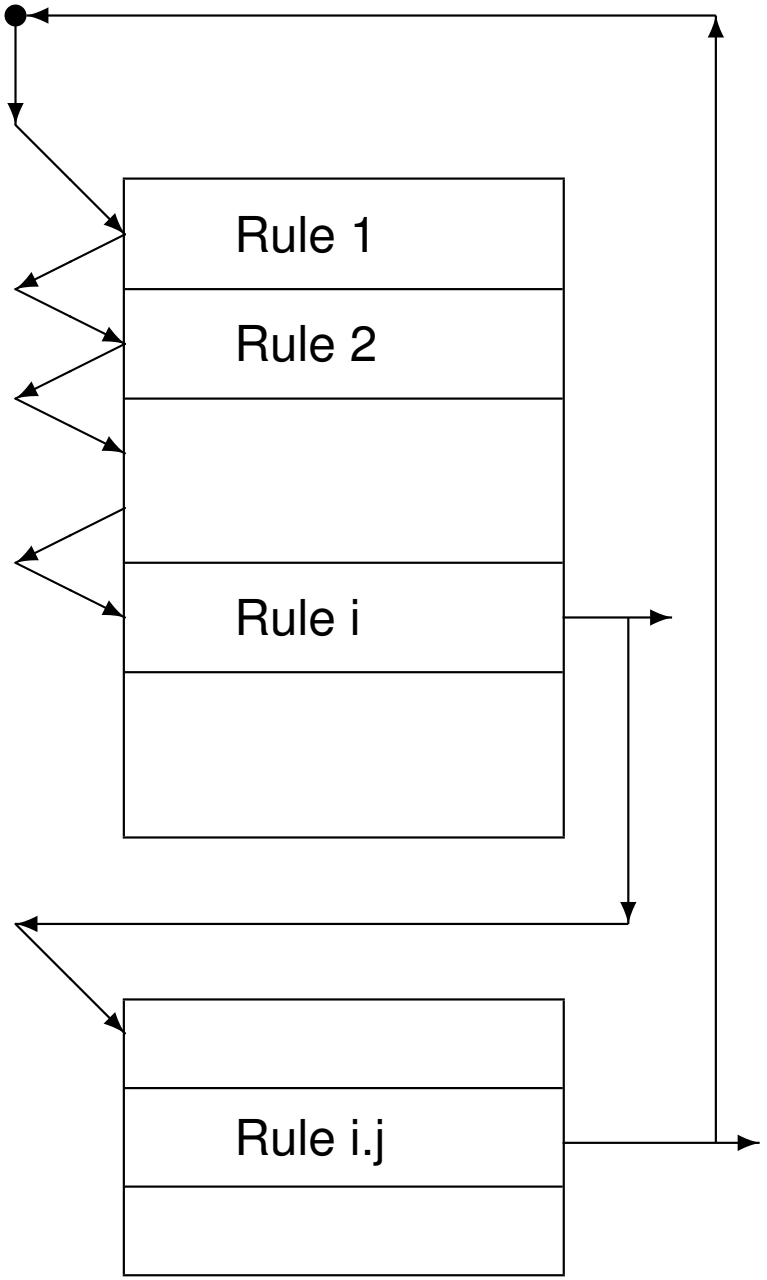
Simple forward-chaining strategies





Forward strategies with switching





Conflict resolution

Let $R = \{r_1, r_2, \ldots, r_n\}$ be a set of rules. The advanced formulation of the conflict resolution task is as follows:

- select a subset $R' \subseteq R$ of rules with satisfied preconditions,
- if there is a unique rule in R' then fire this rule, else
- if there are two or more such rules select one satisfying some auxiliary criteria,
- if R' is empty wait and repeat the cycle.

Basic concepts for conflict resolution

- initial elimination of rules which cannot be applied (refraction),
- selection of a rule which is best, appropriate or likely for performing the current control task,
- *parallel* execution of rules (what does it mean? problems...)

Conflict resolution

Further classification of conflict resolution strategies

- static vs. dynamic strategies; static strategies are based on criteria constant over time, while dynamic ones can take into account current context, time, number of (successful) repetitions of a rule, etc.,
- syntactic vs. semantic strategies; the first one base on the “shape” of the rule preconditions, while the second ones may take into account the current context, desired goal, and evaluable user-specified criteria,
- direct vs. indirect strategies; the direct ones are based on simple comparison of rules and assigned to them “ordering factors”, e.g. priorities, while the indirect strategies can be implemented with an auxiliary knowledge-based system, meta-rules and complex inference schemes,
- strategies based on simple, constant criteria vs. ones modifiable/adaptable and learning.

Conflict resolution

Selected strategies

- rule ordering (a rule appearing earliest has the highest priority),
- data ordering (a rule with the highest priority assigned to its conditions has the highest priority),
- size ordering (a rule with longest list constraining conditions has the highest priority),
- specificity ordering (arrange rules whose conditions are super-set of another rule.
- context limiting (consists in activating or deactivating groups of rules at any time to reduce the occurrence of conflict).

OPS5 and CLIPS strategies

LEX - LEXicographic ordering

- *refraction* (deleting all instantiations fired previously),
- partial ordering based on *recency* (time tags corresponding to the working memory elements used are considered),
- partial ordering with respect to *specificity* (rules with greatest number of tests are considered first),
- finally, arbitrary selection.

MEA – Means-Ends Analysis

- attempts at minimizing the difference between starting and goal state,
- avoid use of recent fact irrelevant to goal achievement,
- considers the recency of the first condition of a rules most important,
- a rule dominating other with respect to specifity or recency of the first condition is selected.

The Rete algorithm

Rete assumptions

- **temporal redundancy** – *most* of the rules have RHS influencing *a few* facts only, and only *a few* rules are affected by those changes,
- **structural similarity** – many rules have a similar pattern in their LHP part.

Rete – principles of work

- avoid matching all the rules agains all the facts at each cycle,
- since rules remain stable and fact changes – facts should find rules (and not rules facts),
- the state of the matching process is save from cycle to cycle,
- the state is recomputed only for changes within the last cycle,
- a network of nodes is organized in the memory,
- one-input nodes perform test on individual facts,
- two-input nodes perform tests accros multiple facts.

Rule-Based Systems

Examples

- OPS5, OPS 83,
- CLIPS,
- JESS,
- Drools,
- G2 (Gensym),
- Sphinx/PC-Shell,
- Prolog,
- BizTalk Rules Engine,
- XpertRules,
- ILOG JRULES,
- Soar.

Logika

Logika

Logika

Logika