4. Dealing with Formalized Knowledge

Textual vs. formal representation of knowledge

Only the formal representation of knowledge allows a computer to interpret that knowledge. This facilitates:

- automatic distribution of knowledge to where it is needed
- support systems making use of that knowledge
- inferencing new knowledge
- finding interrelationships between pieces of knowledge
- automatic detection of inconsistencies

Example of a section of a law:

SchKG 92

The following are exempt from seizure:

⋮

10. the pension and capital sums which are owed or have been paid out to the person concerned or, in the event of his death, to his family as compensation for physical injury or ill-health;

⋮

An ordinary free text search with the following terms would not find that section of law although it highly qualifies:

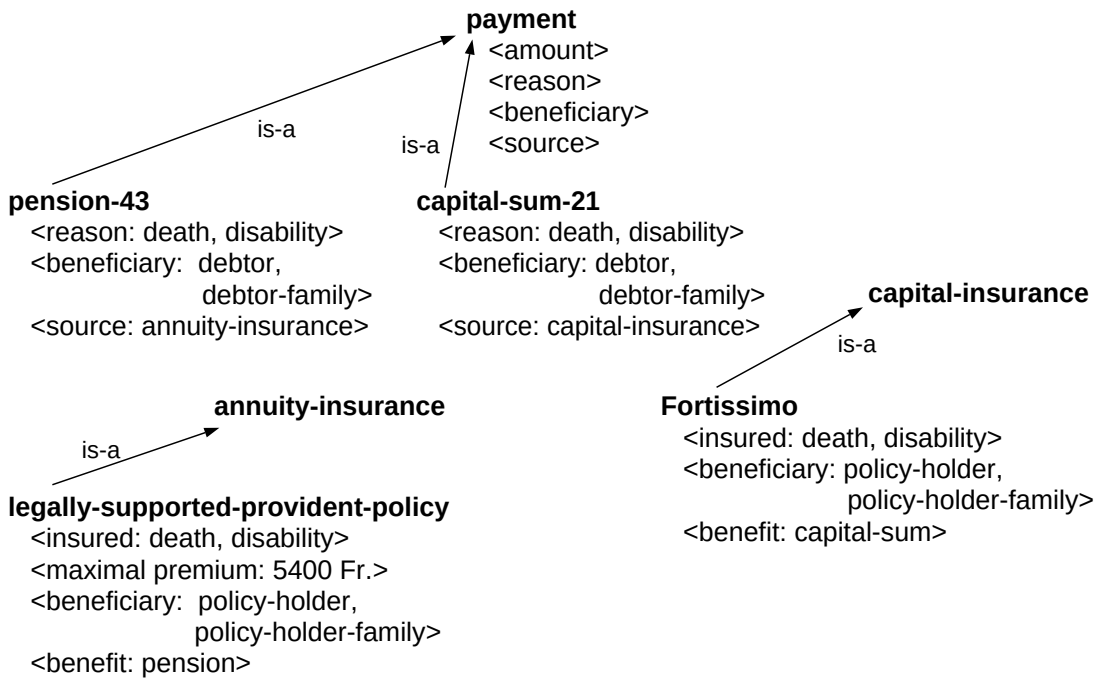
legally supported provident policy,
product Fortissimo

A representation of the contents of the law:

is-bankrupt's-estate(x, y) $\Leftarrow$
  is-bankrupt(x) and
  is-policy-holder-of(x, y) and
  is-life-insurance(y) and not exempt-from-seizure(y)

exempt-from-seizure(x) $\Leftarrow$ pension-43(payment(x)) or
  capital-sum-21(payment(x)) or ...

with the following concept hierarchy:



Definition: Knowledge-Based System

A knowledge-based system has knowledge of its application domain. It needs that knowledge to fulfill its task.

The knowledge is explicitly represented by symbol structures in a dedicated sub-component of the system.

Remark:

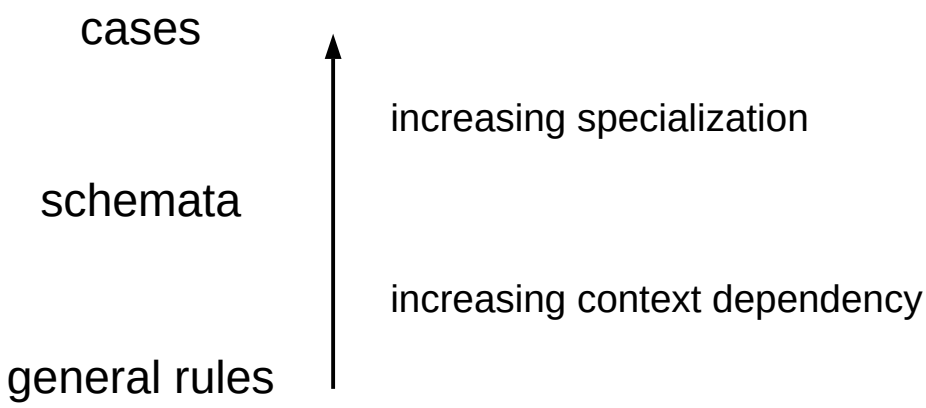
A text is not a representation of knowledge for a knowledge-based system.

Case-based approaches

Definition: Case Base

A *case* is a description of a problem and a solution to it.

A *case base* consists of a set of (indexed) cases.



Case-based reasoning

Uses "lazy generalization" from past similar cases.

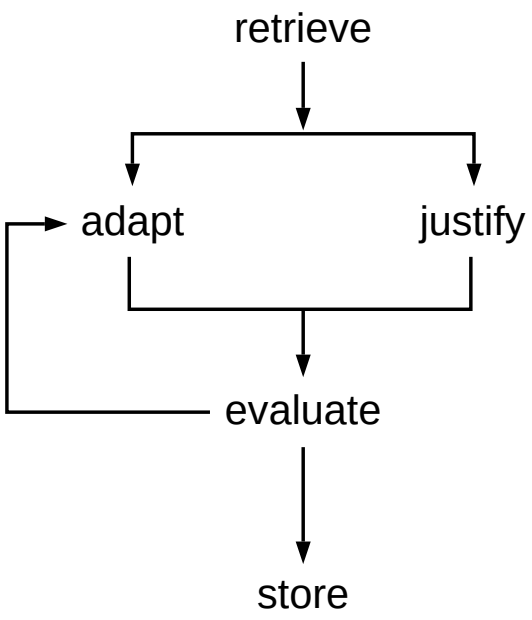
Crucial assumption:

The more similar the current problem p is to a precedent case $c \in CB$, the more similar is the solution of p to the solution of c.

Typical Applications:

- Call centers
- Experience factories
- Best practices

Basic process of case-based reasoning



rule-based reasoning	**case-based-reasoning**
rules are *patterns*	cases are *constants*
retrieved rules match exactly	retrieved cases match partially
rules are applied in an iterative cycle of microevents	cases approximate the entire solution at once, then get adapted and refined
rules are small, ideally independent pieces of domain knowledge	cases are large, redundant chunks of domain knowledge
left-hand sides of rules match part of the problem description, *right-hand sides* give part of the solution or intermediate results	some case attributes match with the problem description, others are part of the solution description

Example of a case description

Problem: document does not print

Symptoms:

"Document printed" in message window

printer is online

printer data led is blinking

printer does not react

Solution: wrong printer type in parameter settings

Case adaptation:

Problem: document does not print

Symptoms:

"Document printed" in message window

printer is online

printer data led is blinking

printer prints empty page

Solution: Adapt from existing case:

wrong parameter settings → wrong filter selected

Example of a case description

Problem: document does not print

Symptoms:

"Document printed" in message window

printer is online

printer data led not blinking

printer is connected

Solution:

wrong printer selected

Case adaptation:

Problem: document does not print

Symptoms:

– same as above –

Solution:

Adapt from existing case:

wrong printer selected → "print to file" box ticked off

Corresponding rule set:

does-not-print(error,printer,application) $\Leftarrow$
  sw-error(error,printer,application) $\vee$ hw-error(error,printer)

sw-error(e,p,a) $\Leftarrow$ application-error(e,p,a) $\vee$ driver-error(e,p,a)

application-error(<u>wrong-printer-type</u>,p,a) $\Leftarrow$
  selected-printer-type(a) $\neq$ type(p)

application-error(<u>wrong-printer</u>,p,a) $\Leftarrow$ selected-printer(a) $\neq$ p

driver-error(<u>wrong-driver</u>,p,a) $\Leftarrow$ $\neg$supports(driver(selected-printer(a)),p)

hw-error(<u>not-connected</u>,p) $\Leftarrow$ $\neg$connected(p)

hw-error(<u>wrong-communication-protocol</u>,p) $\Leftarrow$
  online(p) $\wedge$ connected(p) $\wedge$ $\neg$blinking(data-led(p))

Example of a more structured case description:

1. **When/where to reuse (in which task):**
   Registering evacuees
   context/process: NEO operation
2. **Under which conditions:**
   Hot and humid weather. Location is a tropical country.
3. **What to reuse:** Avoid registration in 3 steps.
4. **Why (originating event):**
   success/failure: It was a failure
   why: It was very time consuming.
   It caused evacuee discomfort.

(after Rosina Weber, http://www.aic.nrl.navy.mil/~weber/)

Reasons to look into a case base:

- maybe there was already developed something similar
- perhaps this has been tried and abandoned (but useful experiences have been made)
- perhaps someone has never tried but showed that it is impossible or much too expensive
- maybe something similar has been done or tried (e.g., not for oil but another fluid, or not a valve but something else which must allow a free flow of viscous fluid)

$\longrightarrow$ lessons learned, best practices, experience factory, etc.

Advantages of case-based reasoning

- natural reasoning process to refer to past, similar cases
- provides inexact matching to retrieve knowlede
- can be used in domains with poor understanding of underlying rules
- provides a starting point and a schema to ask a user appropriate questions to narrow down search
- already allows knowledge provision rather than knowledge searching

Potential disadvantages of formalized knowledge:

- construction and maintenance of formal representations is costly
- less flexibility to changes of the environment
- having automatic systems based on the represented knowledge may reinforce practices that are hard to get changed again

The disadvantages get worse with the degree of formalization.