Rule-based Design Systems

Kazjon Grace
KCDCC
Yet more administrative stuff

- Assessment timetable and course schedule are up on the website: www.arch.usyd.edu.au/~kgra7942/DECO2013
Rule-based Design Systems

- A rule-based design system is comprised of three parts:
  - A set of rules
  - A database of knowledge
  - An algorithm for applying the rules to the knowledge.
A rule is a statement with two parts: an *if-clause* and a *then-clause*.

- If the day is Monday and the time is between 2pm and 5pm.
- Then all the cool kids are in the General Access lab.
Facts

• A fact is a statement of knowledge about something in the world.
  • The day is Monday
  • The time is between 2pm and 5pm

• Facts in the symbolic logic sense have nothing to do with truth, the following are also facts:
  • William Shakespeare’s middle name was Colin
  • The Internet is American
  • Belgian males remove their trousers while driving

(source: Dave’s Web of Lies)
Reasoning algorithms can infer new knowledge from existing knowledge.

The two common algorithms used in rule-based design systems are:

- Forward chaining
- Backward chaining
Forward Chaining

- Forward chaining starts with the available facts and uses the rules to conclude new facts.
  - Each rule is checked to determine whether the if-clause is true according to the known facts
  - If a rule is found, then the statements in the then-clause are added to the knowledge base
  - The process is repeated until no more rules can be fired, or a goal state is reached
Forward Chaining

- Given the following two facts and the single rule, forward chaining can be used to conclude that Kermit is a frog:
  - Kermit is green
  - Kermit can hop
  - If $X$ is green and $X$ can hop then $X$ is a frog
Backward Chaining

• Backward chaining starts with a goal and works backwards to determine if there are facts to support the conclusion that the goal is true.
  • Each rule is searched until one is found with a then-clause that matches a desired goal
  • If the if-clause of a matching rule is not known to be true, then it is added to the list of goals
  • The process is repeated until one of the goals can be shown to be true
Given the following two facts and two rules, backward chaining can be used to answer the question “Can Kermit croak?”

- If X is a frog then X can croak
- If X is green and X can hop then X is a frog
- Kermit is green
- Kermit can hop
• A reasoning algorithm is only ever as good as the knowledge base it has to work with…
**Expert Systems**

- Expert systems use forward and backward chaining to:
  - Analyse information supplied by a user as facts about a specific problem
  - Provide analysis of the problem
  - Recommend a course of action to the user in order to solve the problem
Production Systems

• Production systems use forward chaining to produce new facts from a knowledge base.

• Production systems can be used in design to synthesise structure from behaviour.

• Production systems have been used in cognitive modelling to study the effects of knowledge.
Design Grammars

• Design grammars are production systems that can generate designs according to a set of rules (the grammar).

• A well-defined grammar will generate designs that adhere to design constraints.
Advantages of Design Grammars

- Design grammars give the designer potential to evaluate a large number of alternative designs without laborious work.
- The design produced by a grammar can contain designs that might have been overlooked by a human designer acting unaided.
THE LANGUAGE OF GRAMMARS

• A set of non-terminal symbols
• A set of terminal symbols
• A set of rules, where a rule is of the form:
  • LHS -> RHS
    • LHS and RHS contain terminal or non-terminal symbols
    • LHS must contain at least one non-terminal symbol
**An example grammar**

S -> aSb

S -> ba

Starting with: S

Produces:

- abab
- aababb
- aaababbb
- aaaababbbb
In a design grammar, the symbols produced are interpreted as design elements:

\[ a = \downarrow \]
\[ b = \rightarrow \]

\[ abab = \]
\[ aababb = \]
The production of a design grammar can be interpreted in different ways:

- $a = \text{spring}$
- $b = \text{mass}$

$abab =$

$aababb =$
A REAL DESIGN GRAMMAR

Rules for roof fixing

1. pitched roof
   - Fixing

2. pitched roof
   - Cavity

3. pitched roof
   - Studs

4. pitched roof
   - Plate

5. Place wall plate over single brick wall

6. Place wall plate over outer leaf of double brick wall

7. Place double header over timber studs of framed wall

8. Notch rafters over wall plate

Rules for rafter ends

5. Finish rafter with vertical end.

6. Finish rafter with perpendicular end.

7. Finish rafter with right-angle end.
Context in grammars

• Grammars can be written to be either context-free or context-sensitive.

  • *Context-free grammars do not represent the context of a symbol in the rules*
  • *Context-sensitive grammars represent the context of a symbol using a prefix and/or suffix*

• Context sensitivity can be very important to distinguish between similar structures in order to do the right thing.

  • *Time flies like an… arrow*
  • *Fruit flies like a… banana*
Applying the rules of a grammar

• How the rules are applied has a significant impact on what can be produced.
  • Rules applied in sequence (one symbol replaced at a time)
    • Chomsky grammars - structured designs
  • Rules applied in parallel (all possible symbols replaced at once)
    • Lindenmayer grammars - trees, flowers, fractals, organic forms
**Deterministic vs Non-deterministic**

- Some rule interpretation systems can support non-deterministic firing of rules, e.g.: L-Systems

- Multiple rules can be specified for the same LHS and the rule fired is chosen at random.

- Some systems allow the different rules applying to one situation to be weighted and fired with different probabilities.