COMP 4200: Expert Systems

Dr. Christel Kemke
Department of Computer Science
University of Manitoba

CLIPS Introduction 1

Overview and Introduction
- CLIPS Programming System
- Basic CLIPS Constructs and Syntax
  - Basic Data Types
  - Basic Data Structures
  - Basic Condition Patterns
  - Rules

CLIPS – Background

CLIPS – C Language Integrated Production System
- Production System refers to Rule-Based Systems
- Originally developed by NASA
- download, user's manual, developer's forum etc. see CLIPS link on course web-page
- CLIPS is free, shareware; can even be used for commercial applications

CLIPS – Programming Systems

CLIPS Editor
- load, save and edit CLIPS program files

CLIPS Interpreter
- enter commands, execute CLIPS programs

Execution-Menu
- set execution parameters (e.g. watch)

Browse-Menu
- manage constructs

CLIPS - Example

CLIPS> (load "file") or use file-menu
CLIPS> (assert (today is Tuesday))
f-0 (today is Tuesday)
CLIPS> (retract 0) retract fact 0
CLIPS> (facts) display current facts
CLIPS> (reset) reset facts; re-run definitions
CLIPS> (clear) clear CLIPS Interpreter (everything's gone now)

CLIPS – Defining Facts

CLIPS> (deffacts today *defines the set* today
(today is Tuesday) *with 2 facts*
(weather is cold))

CLIPS> (reset) *re-read definitions*

CLIPS> (facts) *display current facts*
f-0 (initial-fact) *system "start-fact"*
f-1 (today is Tuesday)
f-2 (weather is cold)
For a total of 3 facts.

7

Working with Facts

Changing the Fact Base

- Adding facts (assert <facts>)
- Deleting facts (retract <fact-identifier>)
- Modifying facts (modify <fact-identifier> (<slot-name> <slot-value>))
- Duplicating facts (duplicate <fact-identifier> (<slot-name> <slot-value>))

Monitoring fact base

- Print all facts (facts)
- Displays changes (watch facts)

8

CLIPS – Basic Constructs

Basic Language Elements:

- Fields (basic Data Types)
- Facts (used in condition patterns)
- Rules (condition-action rules)
- Templates (like records; define facts)
- Classes (like objects; define facts)
- Messagehandlers (like methods defined for classes; used in condition patterns and actions)

9

CLIPS – Basic Syntax

Syntax of Constructs is LISP-like, e.g.

- (deffacts <fact-list-name> <fact-list>)
- (defrule <rule-name> <rule-body>)
- (defclass <class-name> <class-body>)

General form:

(function-name parameter-list)

Don't forget the brackets (...) !

10

CLIPS – Rules in General

In CLIPS (and other Rule-based Languages) Rules are the core elements of program execution.

Rules consist of a Condition-part and an Action-part.

```
(defrule rule-name “Comment”
        <condition patterns>
 =>
        <actions>)
```


11

CLIPS – Rules in General

The condition-part of rules consists of patterns which are matched against facts; facts are defined based on data structures (like templates and classes); plus additional operators to form complex patterns.

The action-part contains commands for modifying facts or the fact base, or external actions like printout or load.

12

CLIPS – Patterns and Rules

CLIPS provides a rich set of mechanisms to define and describe patterns used as Conditions ("Condition Elements") in rules.

examples:

- simple facts;
- facts based on templates;
- facts based on classes (instances);
- facts with variables and wildcards;
- connections of facts (logical connectives like 'and' and 'or')

13

CLIPS – Basic Data Types and Constructs

CLIPS

- Fields (data types)
- Facts
- Templates
- Rules

14

CLIPS – special syntax for fields

CLIPS is case-sensitive:

symbols Space different from space
but space
same as _space

strings "_space" different from "space"

special characters in strings: insert \

"\"symbol\"" → "symbol" as string

"\\symbol\\"→ \symbol\ as string

15

CLIPS - Fields

Fields (data types)

- float real number
- integer integer numbers
- symbol character-sequence (without special characters); ends with space
- ? and $? not at beginning of a symbol
- ? and $? are pre-fix of variables
- string character-sequence in double-quotes
- instance name name of an instance in [ ]

16

Fields - Examples

Fields (data types)

- float 4.00, 2.0e+2, 2e-2
- integer 4, 2, 22
- symbol Alpha24*, !?@*$
- string "Johnny B. Good"
- instance name [titanic], [PPK]

Variables

?var, ?x, ?day variables for single field value

$?names variable for multi-field value

17

CLIPS –Facts

Facts

- a relation-name,
- an ordered sequence of values (ordered facts), or
- a set of (slot-name slot-value)-pairs (i.e. deftemplate-facts)

examples:

(today is Thursday)
(person (name "Johnny B. Good") (age 25))

18

Ordered Facts

Ordered facts

- are facts defined without (explicit) template;
- the field-values are ordered.

Examples:

(number-list 1 2 55 6 7 42)
(today is Sunday)

19

Deftemplate Facts

Deftemplate-facts

- are facts defined based on a template;
- slots can be arranged arbitrarily, there is no specific order.
- Define a template for describing a set of facts using deftemplate (record structure) .
- Use deffacts to create a list of facts based on a template.

20

CLIPS – Defining Templates

Templates

- keyword 'deftemplate'
- relation-name
- list of slot-specifications: (slot slot-name)...

Example:

```
(deftemplate person
    (slot name)
    (slot age))
```

You can add **type constraints** to the slots, e.g.

```
    (slot name) (type STRING)
    (slot age) (type INTEGER)
```


21

No Order in Deftemplate-Facts

Slots in templates can be arranged arbitrarily. The following templates define the same data structure:

```
(deftemplate person          (deftemplate person
        (slot name)
        (slot age)
        (slot age))
        (slot name))
```

The template-facts

```
(person (name Johnny) (age 42))
        (person (age 42) (name Johnny))
```

are regarded the same (→ pattern matching).

22

CLIPS – Defining a Fact List

Fact List

- keyword 'deffacts'
- name of the fact list
- list of facts, each fact in (...)

Example 1:

Define the set today with 2 facts:

```
(deffacts today
        (today is Thursday)
        (weather is cold)
)
```


23

CLIPS – Defining a Fact List

Example 2:

Define a set of old-students based on a student-template:

```
(deftemplate student
        (slot name)
        (slot age))
```

```
(deffacts old-students
    (student (name "Igor") (age 50))
    (student (name "Berta") (age 80)))
```


24

CLIPS – Defining Rules

Rules

- keyword 'defrule'
- name of the rule
- condition patterns (condition elements)
- =>
- actions (+ control structure)

```
(defrule rule-name "Comment"
        <condition patterns>
    =>
        <actions>)
```

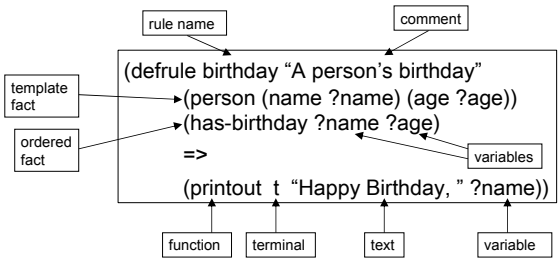
25

Defining Rules - Example

```
(defrule birthday "A person's birthday"
      (person (name ?name) (age ?age))
      (has-birthday ?name ?age)
=>
      (printout t "Happy Birthday," ?name))
```


26

Rule - Example



27

CLIPS - Refraction

refraction: rule fires only once on the same data
refresh (or reset): rule starts all-over again

```
(defrule birthday "A person's birthday"
     (person (name ?name) (age ?age))
     (has-birthday ?name ?age)
      =>
     (printout t "Happy Birthday," ?name))
```

question: How many times does the rule above fire and produces an output when the fact-base consists of the 'old-students' fact-list from above?

28

Rule - Test

```
(defrule birthday "A person's birthday"
      (person (name ?name) (age ?age))
      (has-birthday ?name ?age)
      =>
      (printout  t  "Happy Birthday, " ?name))
```

exercise: Test the rule above and then modify it in such a way that it produces outputs like "Mary is turning 40 today."