

A Tutorial on PAL (Protégé Axiom Language)



Samson Tu
Stanford Medical Informatics
Stanford University

2001 Fifth International Protégé User
Group Meeting
Newcastle upon Tyne

(With some materials taken from William
Grosso's presentation at 1999 Fourth
International Protégé User Group Meeting)

Goals of Tutorial

- ◆ **Overview**
 - What PAL is and what it can be used for
 - How PAL is integrated into the Protégé-2000 framework
- ◆ **Mechanics**
 - How to write PAL constraints
 - How to use PAL constraints and query tabs
 - How to debug PAL constraints
- ◆ **How I (mis)use PAL constraints and queries in my application**

PAL: What's It?

◆ What it is:

- A constraint language (beyond slot facets) that helps to enforce the semantic properties of knowledge bases encoded in Protégé
- A query language for searching instances that satisfy certain relationships

◆ What it is not:

- A general predicate-logic language
- A way to do logic programming in Protégé-2000
- Another way to write definitions of concepts modeled in Protégé-2000

3

Example

◆ Subordinates of editors make less than their supervisors

```
(defrange ?empl :FRAME Employee responsible_for)
(defrange ?edit :FRAME Editor)
(forall ?edit
  (forall ?empl
    (=> (and (own-slot-not-null salary ?edit)
             (own-slot-not-null salary ?empl)
             (responsible_for ?edit ?empl))
        (< (salary ?empl) (salary ?edit))))))
```

4

A Limited Predicate Logic Extension of Protégé-2000

- ◆ We decided on a variant of KIF
- ◆ We use the KIF connectives and the KIF syntax
- ◆ Not all the KIF constants and predicates are included
 - e.g. theory of arithmetic is much smaller
- ◆ (defrelation ...), (deffunction...) are omitted

5

Constraints and Axioms

- ◆ Use the syntax of logic but have different semantics
 - Axioms can be used to assert new knowledge
 - Constraints are restrictions on existing knowledge
- ◆ (forall ?x (exists ?y (rel-name ?x ?y)))
 - Asserted as an axiom: it's reasonable to create a skolem constant and bind it to ?y
 - Asserted as a constraint: constraint is violated if no existing instance ?y satisfies the relation
- ◆ PAL: Protégé ~~Axiom~~ Constraint Language

6

Model-checking, rather than theorem proving

- ◆ Make strong “closed world” assumptions
- ◆ Main goals:
 - Detect incomplete entry of information
 - Check entered information for inconsistencies

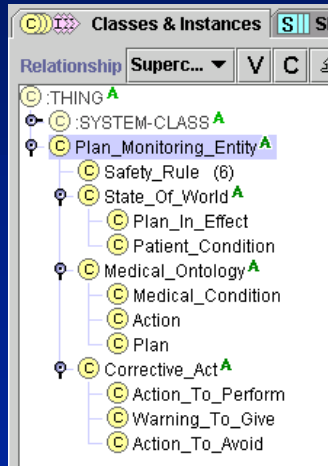
7

Goals of Tutorial

- ◆ Overview
 - What PAL is and what it can be used for
 - How PAL is integrated into the Protégé-2000 framework
- ◆ Mechanics
 - How to write PAL constraints
 - How to use PAL constraints and query tabs
 - How to debug PAL constraints
- ◆ How I (mis)use PAL constraints and queries in my application

8

Example Ontology



- ◆ Simple ontology to model safety rules in oncology

- P. Hammond et al. *Safety and Decision Support in Oncology* Meth Inform Med 1994(33) 371-81

- ◆ Safety rules to be written as PAL constraints later

9

Compatibility with Protégé-2000 Framework

- ◆ PAL constraints are themselves frames
- ◆ Integration with frame-based knowledge model
- ◆ PAL Constraint tab and PAL Query tab allow editing, validation, and execution of constraints and queries
- ◆ Constraint and Query Engines are plugins
- ◆ PAL API allows evaluation of constraints and queries by applications

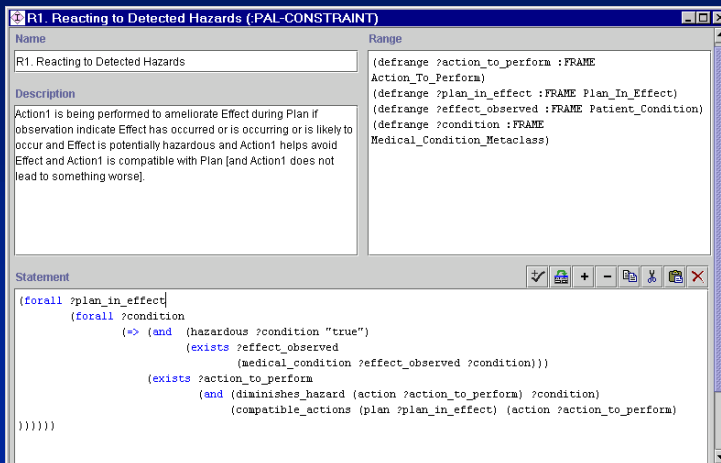
10

Constraints as Frames

- ◆ Constraints are reified as frames
 - Acquiring a constraint is really “acquiring an instance of :PAL-CONSTRAINT” class
 - To a simple KB server, a constraint is just another frame
- ◆ :PAL-CONSTRAINT slots
 - name: a short label
 - description: Natural-language documentation
 - range: declarations of quantified variables
 - statement: constraint written in PAL

11

Acquiring a PAL Constraint in Protégé-2000



The screenshot shows a window titled "R1. Reacting to Detected Hazards (PAL-CONSTRAINT)". The window is divided into three main sections: Name, Description, and Statement. The Name field contains "R1. Reacting to Detected Hazards". The Description field contains a natural language description: "Action1 is being performed to ameliorate Effect during Plan if observation indicate Effect has occurred or is occurring or is likely to occur and Effect is potentially hazardous and Action1 helps avoid Effect and Action1 is compatible with Plan [and Action1 does not lead to something worse]". The Range field contains a list of frame declarations: "(defrange ?action_to_perform :FRAME Action_To_Perform)", "(defrange ?plan_in_effect :FRAME Plan_In_Effect)", "(defrange ?effect_observed :FRAME Patient_Condition)", and "(defrange ?condition :FRAME Medical_Condition_Metaclass)". The Statement field contains a PAL constraint written in code: "[forall ?plan_in_effect] [forall ?condition] (=) (and (hazardous ?condition 'true') (exists ?effect_observed (medical_condition ?effect_observed ?condition))) (exists ?action_to_perform (and (diminishes_hazard (action ?action_to_perform) ?condition) (compatible_actions (plan ?plan_in_effect) (action ?action_to_perform))))))".

12

Integration with Protégé-2000 Knowledge Model

- ◆ Quantified variables are declared as holding instances of a class
 - (defrange ?action :FRAME Action)
 - (forall ?action ...)
 - (exists ?action ...)
- ◆ Slots are predicates
 - (diminishes_hazard ?action dehydration)
- ◆ Cardinality-single slots are functions
 - (label ?action) returns a string

13

New functions and predicates are implemented procedurally

- ◆ KIF has the (*defrelation ...*) and (*deffunction*) constructs to define new relations and functions
- ◆ PAL supports classes and slots as unary and binary relations, and single cardinality slots as functions
- ◆ Additional relations (e.g., >, subclass_of) and functions are defined procedurally

14

PAL Query Language

- ◆ Taking a constraint and finding instances that violate it is much like finding instances that satisfy a statement
 - (not (exists ?x (...)))
- ◆ Query engine introduces two keywords
 - find
 - findall
- ◆ Query engine limited to finding instances of one variable (for simplicity)

17

Goals of Tutorial

- ◆ Overview
 - What PAL is and what it can be used for
 - How PAL is integrated into the Protégé-2000 framework
- ◆ Mechanics
 - How to write PAL constraints
 - How to use PAL constraints and query tabs
 - How to debug PAL constraints
- ◆ How I (mis)use PAL constraints and queries in my application

18

How to Write PAL Constraints

- ◆ Learn the syntax and available predicates and functions
- ◆ Write down constraints in NL
- ◆ Develop/learn/modify ontology
- ◆ Declare variables
- ◆ Write constraint
- ◆ Check syntax
- ◆ Try out examples
- ◆ Iterate

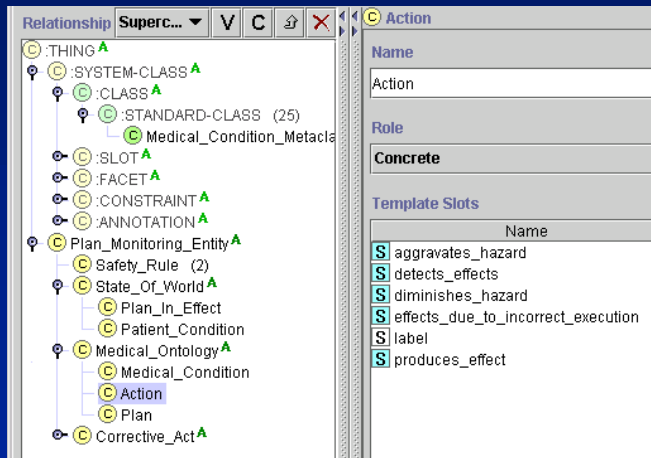
19

Example Constraint in NL

- ◆ **Action1** is being performed to **ameliorate** **Effect2** during **Plan3** if **observation** indicates **Effect2** has occurred or is occurring or is likely to occur and **Effect2** is potentially **hazardous** and **Action1** helps to **avoid** **Effect2** and **Action1** is **compatible** with **Plan3**.

20

Learn/Modify Ontology



21

Declaring Variables

- ◆ Quantified variables refer to frames
- ◆ Syntax for explicit declaration
 - *(defrange ?action :FRAME Action)*
- ◆ Attached constraints
 - A variable (?foo) used in a constraint needs explicit range declaration only if the constraint is not attached to a class
 - If a constraint is attached to a class, then by default undeclared variables range over instances of that class

22

Write Constraint

```
(forall ?plan_in_effect
  (forall ?condition
    (=> (and (hazardous ?condition "true")
             (exists ?effect_observed
              (medical_condition ?effect_observed ?condition))))
    (exists ?action_to_perform
      (and (diminishes_hazard (action ?action_to_perform)
                               ?condition)
           (not (incompatible_actions (plan ?plan_in_effect)
                                       (action ?action_to_perform)
                                       ?condition))))))
))))))
```

Do Demo

23

Exercise

◆ Write the following as PAL constraints:

- No plan in instances of Plan_In_Effect contains an action that is hazardous
- There is an action (Action1) to perform if there is another action (Action 2) in Plan and Action 2 produces Effect and Effect is potentially hazardous and Action 1 helps avoid Effect and Action 1 is compatible with Plan

24

Debugging PAL Constraints

- ◆ Use tools that help with balancing of parenthesis and with indentations
 - Anton's PAL editor does these and more!
- ◆ Try small examples
- ◆ Test components
- ◆ Trace built-in predicates and functions
- ◆ Write existence clauses as queries

25

Goals of Tutorial

- ◆ Overview
 - What PAL is and what it can be used for
 - How of PAL is integrated into the Protégé-2000 framework
- ◆ Mechanics
 - How to write PAL constraints
 - How to use PAL constraints and query tabs
 - How to debug PAL constraints
- ◆ How I (mis)use PAL constraints and queries in my application

26

Uses of PAL as KB Integrity Constraints

- ◆ Constraints that define consistency conditions among classes and instances in knowledge base

- e.g., subclasses of `Medical_Condition` class should be all instances of `Medical_Condition_MetaClass`

```
(defrange ?subclass :FRAME :STANDARD-CLASS)
```

```
(=> (subclass-of ?subclass Medical_Conditions_Class)  
(instance-of ?subclass Medical_Conditions_MetaClass)  
)
```

27

Use of PAL as Application Criteria Language

- ◆ Application loads runtime data (e.g., prescribed medications of specific patients) as instances in knowledge base
- ◆ Guideline interpreter invokes constraint engine (through API) to evaluate "constraints"
- ◆ Guideline interpreter maps "constraint violation" or "constraint satisfaction" to truth values

28

Example: PAL as Application Criteria Language

- ◆ There is a prescribed medication to which the patient has adverse reaction

- (\$patient_id is a variable whose value is instantiated by the application. It's a constant string for PAL constraint engine)

```
(exists ?current_med
  (and (patient_id ?current_med $patient_id)
    (exists ?adverse_reaction
      (and (patient_id ?adverse_reaction $patient_id)
        (or (= (drug_name ?current_med
              (substance ?adverse_reaction))
            (subclass-of (drug_name ?current_med)
              (substance ?adverse_reaction)))))))))
```

29

Use of PAL as Application Query Language

- ◆ Application loads runtime data (e.g., prescribed medications of specific patients) as instances in knowledge base
- ◆ Guideline interpreter invokes query engine (through API) to query the knowledge base
- ◆ Guideline interpreter uses query result in problem solving

30

Example: Use of PAL Query

- ◆ Find a patient's anti-hypertensive medications that are not ACE Inhibitors

- potential candidates for drug substitution

```
(findall ?med
  (and (patient_id ?med $patient_id)
    (not (subclass-of (coerce-to-class (drug_name ?med))
      (coerce-to-class "ACE_Inhibitors"))))
    (subclass-of (coerce-to-class (drug_name ?med))
      (coerce-to-class "Anti-Hypertensive_Drugs"))))
```

31

What is a knowledge base ?

- ◆ Used to be classes and instances

- ◆ Now also includes constraints and queries

- Instances with an "interpretation" beyond the standard meaning associated to frames
- Custom pieces of java code that implement new relations (possibly domain specific) for the constraint language
- Knowledge base holds expressions that are evaluated and used in problem solving.

32