FORMAL SPECIFICATIONS USING PDDL

- ✓ Introduction
- ✓ WordPad
- ✓ PDDL

Formal Specification – INTRO

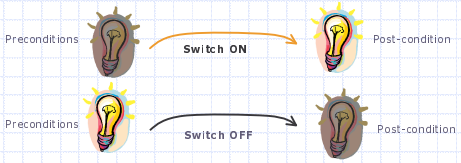
We can describe a system using specification languages.

Here we see how PDDL can be used to describe a system.

PDDL primarily describes a system using a set of preconditions and post-conditions.

PDDL = Planning Domain Definition Language

Formal Specification – BULB



States – ON, OFF

Actions – Switch ON, Switch OFF

Formal Specification – BULB

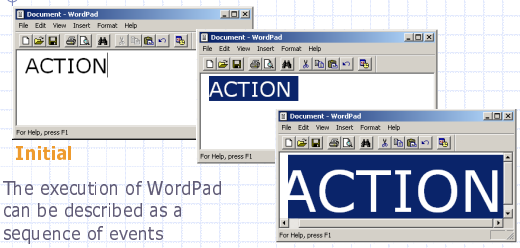
States – ON, OFF
Actions – Switch ON, Switch OFF

Switch ON:
- Precondition: OFF — The state of the system before this action can be applied
- Effect: ON — The state of the system after applying this action

Switch OFF:
- Precondition: ON
- Effect: OFF

WordPad

WordPad – Event Sequence



Initial

The execution of WordPad can be described as a sequence of events

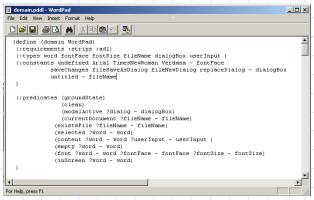
An event is an user input that causes WordPad to transit from one state to another

Final

WordPad – Formal Specification

We can describe the event sequence of WordPad using a specification language.

It can be executed to verify the correctness of the specification



WordPad – Formal Specification

Each state of WordPad can be considered to be the collection of its visible and invisible contents

Taskbar...Toolbar...

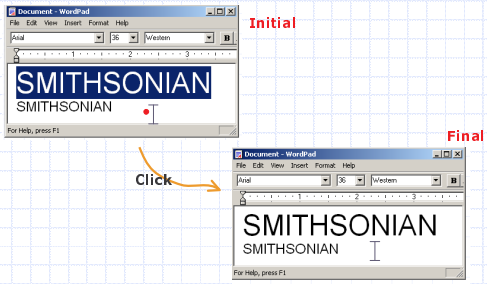
Outside view port

Events induce transition from one state to another

We can specify each event in WordPad formally using a specification language such as PDDL

Let us see some examples...

WordPad – Example



WordPad – Example

Action Name

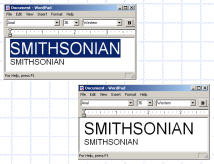
- Deselect all selected words

Precondition

- No dialog boxes open

Effect

- No words are selected



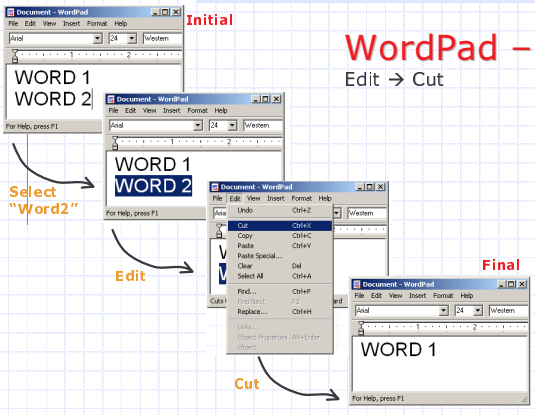
WordPad – Example

PDDL

```
(:action Click-To-Deselect
:precondition (groundState)
:effect (forall (?word - word) (not (selected ?word)) )
)
```

This indicates that for all the word that are defined in the system, they should **not** have the property **selected**

WordPad – Edit → Cut



WordPad – Edit → Cut

Action Name

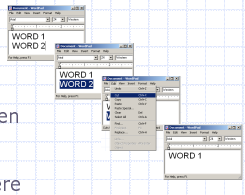
- Edit → Cut

Preconditions

- No dialog boxes are open

Effect

- All those words that were selected are deleted from the view
- No words are selected



PDDL

http://www.ida.liu.se/~pahas/maip/writing.html
http://www.informatik.uni-freiburg.de/~koehler/aips/PDDL-MANUAL.ps.gz

PDDL – Planning Domain Definition Language

It is a domain definition language which is supported by most planners.

Used to define the properties of a domain
- **predicates** to be used
- **action** definition

Example of planners
- IPP - http://www.informatik.uni-freiburg.de/~koehler/ipp.html
- UCPOP

PDDL – Predicates

Predicate defines the **property** of an **object**, which can be TRUE of FALSE

Example:

- **Cloudy**
- **Big**
- **Yellow**

Use in PDDL

Example:

- **Yellow T-Shirt**

PDDL – Variables / Types

C

- int iterator;
- myStruct bigStructure;

PDDL

- ?iterator – int
- ?bigStructure - myStruct

In PDDL data types are not predefined

:types int myStruct

PDDL – not and or

```
(and  ( Yellow T-Shirt )
      ( Big Shoes )
)

( not ( Yellow Shoes )
)

( and ( Yellow T-Shirt )
      ( not ( Yellow Shoes ) )
)
```

PDDL – forall

Let the all T-Shirts in this world be Yellow

```
(:types T-Shirt)
(:predicates (Yellow ?things – T-Shirt)
)
...
(forall (?things – T-Shirt) (Yellow ?things)
)
```

This implies that the **property** Yellow should be true for all **objects** in the domain that are of type T-Shirt

PDDL – exists

If there exists even one Yellow shoe…

```
(:types Shoes)
(:predicates (Yellow ?things – Shoes)
)
...
(exists (?things – Shoes) (Yellow ?things)
)
```

This evaluates to TRUE if there exists one or more **objects** which has the **property** Yellow.

PDDL – Domain Definition

- ✓Requirements – packages to be used
- ✓Types – user defined types
- ✓Constants – constant to be used in this domain
- ✓Predicates – definition of truth statements
- ✓Action - operators
  - Preconditions – predicates that must be **TRUE** before this operator is applied
  - Effects – predicates that become true after this operator is applied

PDDL – Problem Definition

Define the problem to be solved

- ✓Initial State – define predicates which are true at the beginning of the problem
- ✓Goal State - define predicates which are true at the end of the problem

PDDL – Domain Definition - **Syntax**

```
(define (domain DOMAIN_NAME)
 (:requirements [:strips] [:equality] [:typing] [:adl])
 (:types TYPE_1  TYPE_2  ...  TYPE_N)
 (:predicates (PREDICATE_1  [?A1  ?A2  ...  ?AN])
              (PREDICATE_2  [?A1  ?A2  ...  ?AN])
              ...)

 (:action ACTION_1
   [:parameters (?P1 ?P2 ... ?PN) ]
   [:precondition PRECOND_FORMULA]
   [:effect EFFECT_FORMULA]
 )

 (:action ACTION_2
 ...)
```

PDDL – Domain Definition - **Precondition**

STRIPS domain

```
(PREDICATE_NAME  ARG1 ... ARG_N)
(and  ATOM1  ...  ATOM_N)
```

ADL domain (***in addition***)

```
(not  CONDITION_FORMULA)
(and  CONDITION_FORMULA1 ... CONDITION_FORMULA_N)
(or   CONDITION_FORMULA1 ...  CONDITION_FORMULA_N)

(forall (?V1 ?V2 ...)  CONDITION_FORMULA)
(exists (?V1 ?V2 ...)  CONDITION_FORMULA)
```

PDDL – Domain Definition - **Effects**

STRIPS domain

(PREDICATE_NAME ARG1 … ARG_N)
(**not** (PREDICATE_NAME ARG_1 … ARG_N))
(**and** ATOM1 … ATOM_N)

Can we have:
(**or** ATOM1 … ATOM_N)

ADL domain (***in addition***)

(**when** CONDITION_FORMULA EFFECT_FORMULA)
(**forall** (?V1 ?V2 …) EFFECT_FORMULA)

Can we have:
(**exists** (?V1 ?V2 …) EFFECT_FORMULA)

PDDL – Domain Definition - **Types**

When using a parameter or bound variables in an action you must indicate its type

(**:types dialogBox word**)
…
:parameters ?db – **dialogBox**
:effect (**forall** (?w – **word**) selected ?word)

PDDL – Problem Definition

define (**problem** PROBLEM_NAME)
(**:domain** DOMAIN_NAME)
(**:objects** OBJ_1 OBJ_2 … OBJ_N)
(**:init** ATOM_1 ATOM_2 … ATOM_N)
(**:goal** CONDITION_FORMULA)
)

DEMO