General approach to automation of FLASH subsystems

Bogusław Kosęda

Agenda

- Motivation
- Nature of the problem and underlying information
- Required capabilities
- Basic formalization of problem domain
- Anatomy of the solution

Motivation

Necessity of uniform approach to design and implementation of automation software for high energy experiments.

Ultimate role of the automation software:

- Maximization of lasers availability.
  - Automation of routine activities as startup, shutdown ...
  - Continuous monitoring of hardwares condition.
  - Human error minimalization.
  - Autorecovery from specific trips.
- Improvement of breakdown cause location.

Important software engineering challenges:

- Standardization of design, implementation and documentation of automation software for laser subsystem.
- Development of formal verification and testing procedures, thereby improvement of reliability, predictability and safety of the software.

From the outset: underlying information

- Online measurements and status signals form the hardware and their meaning
- Definitions of operation modes of the subsystem
- Definitions of known fault patterns and remedial measures
- Primitive procedures and actions that are primary tools of the automation

Basic formalization

- Observables:
  Input DOOCS signals of types int,bool,symbolic.
- Quantized condition:
  Symbolic variable with limited domain. Its value is based on values of the observables
- State of the subsystem:
  Valuation of all quantized conditions
- Operation mode:
  Valuation of set of quantized conditions
- Procedure:
  Procedure + valuations that its execution entails
- Long run sequence:
  Sequence of procedures bringing the subsystem from current to the desired operation mode.
- Fault pattern:
  Valuation of set of observables
- Exception:
  Classified fault pattern with ascribed remedial procedure.

Automation capabilities

Required capabilities:

- To automate achieve and maintain pre-selected operation mode
- To adapt to unpredictably changing conditions of the hardware
- To recover the subsystem from known faults.
- To persist in above processes to reasonable extent

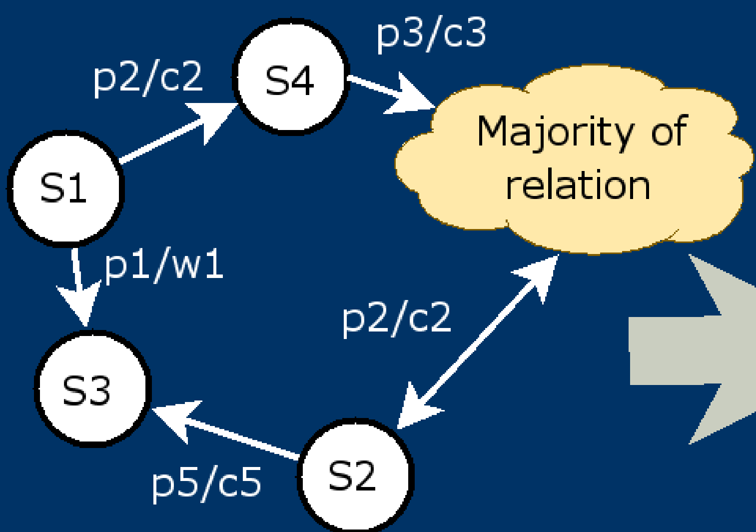
Anatomy of the solution - planner

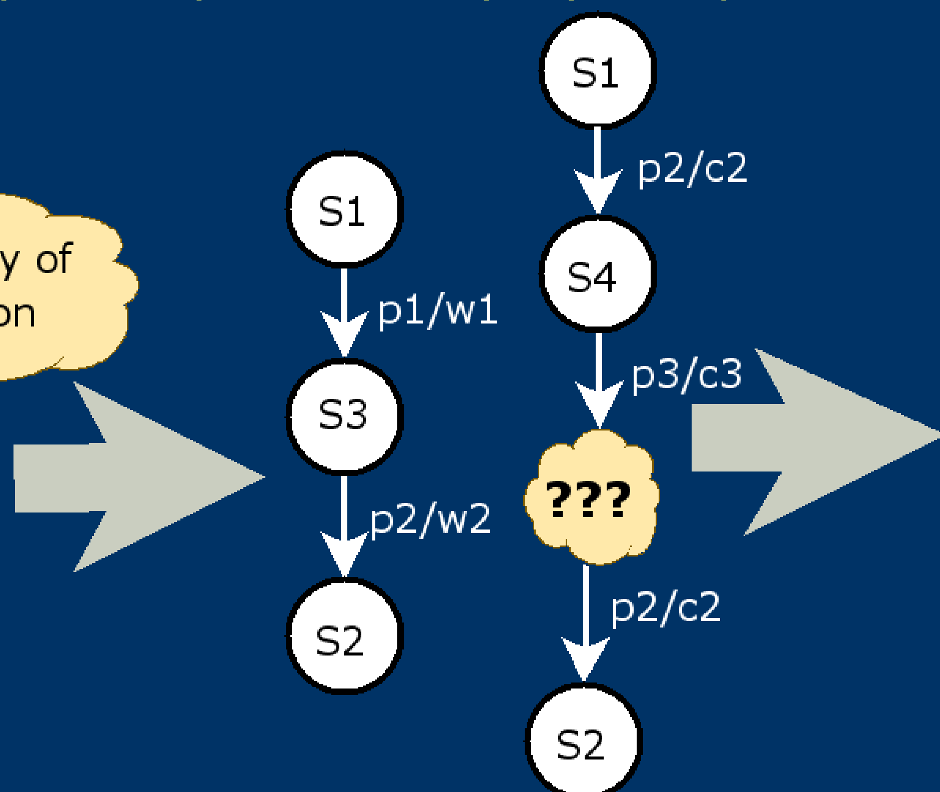
Ensemble of:
- State estimator
- Transition relation (state space)
- Planning (path finding) algorithm
- Execution engine

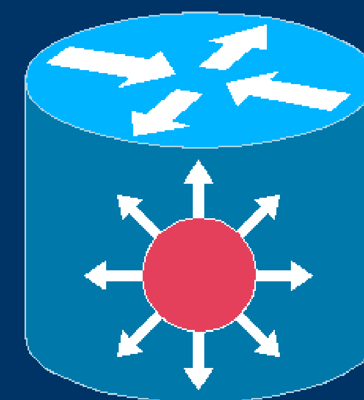
est_st(S1)
planseq(S1,S2)
exec(p1)
est_st(S1)
planseq(S3,S2)
exec(p2)

planseq(S1,S2)={p1,p2,...,pn}

ss(S1,S2)

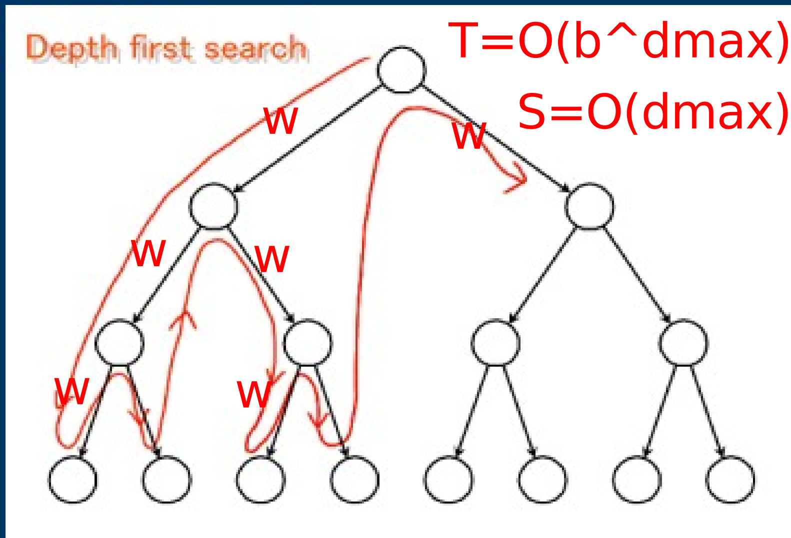


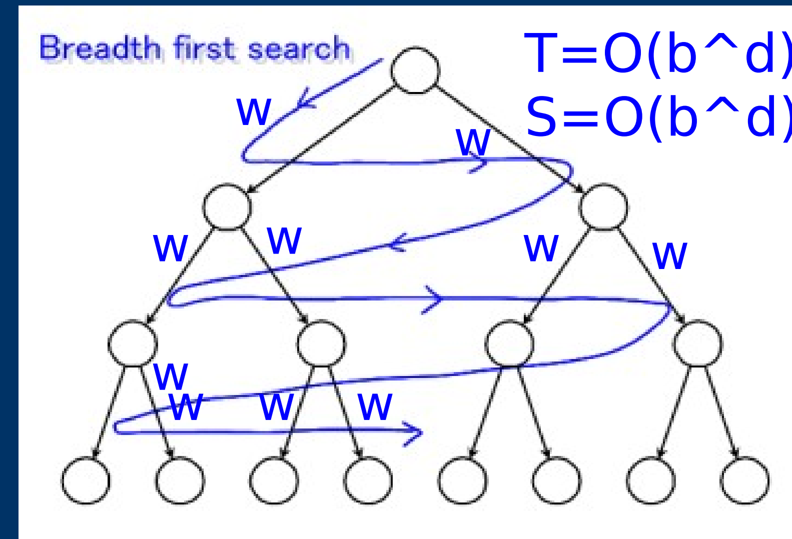




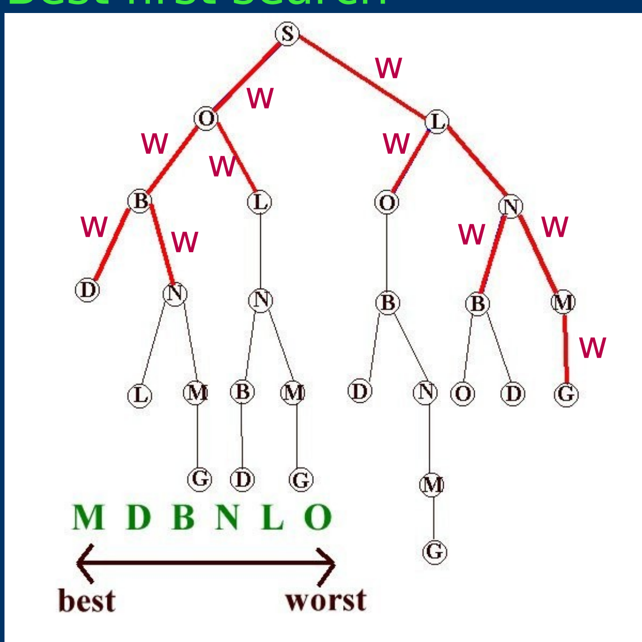
Planner execution engine

Planner's algorithms





Best first search



$f(n)=wi+h(n)$
$h(n)=w1+w2+\ldots$

T=O(b^d)
S=O(b^d)

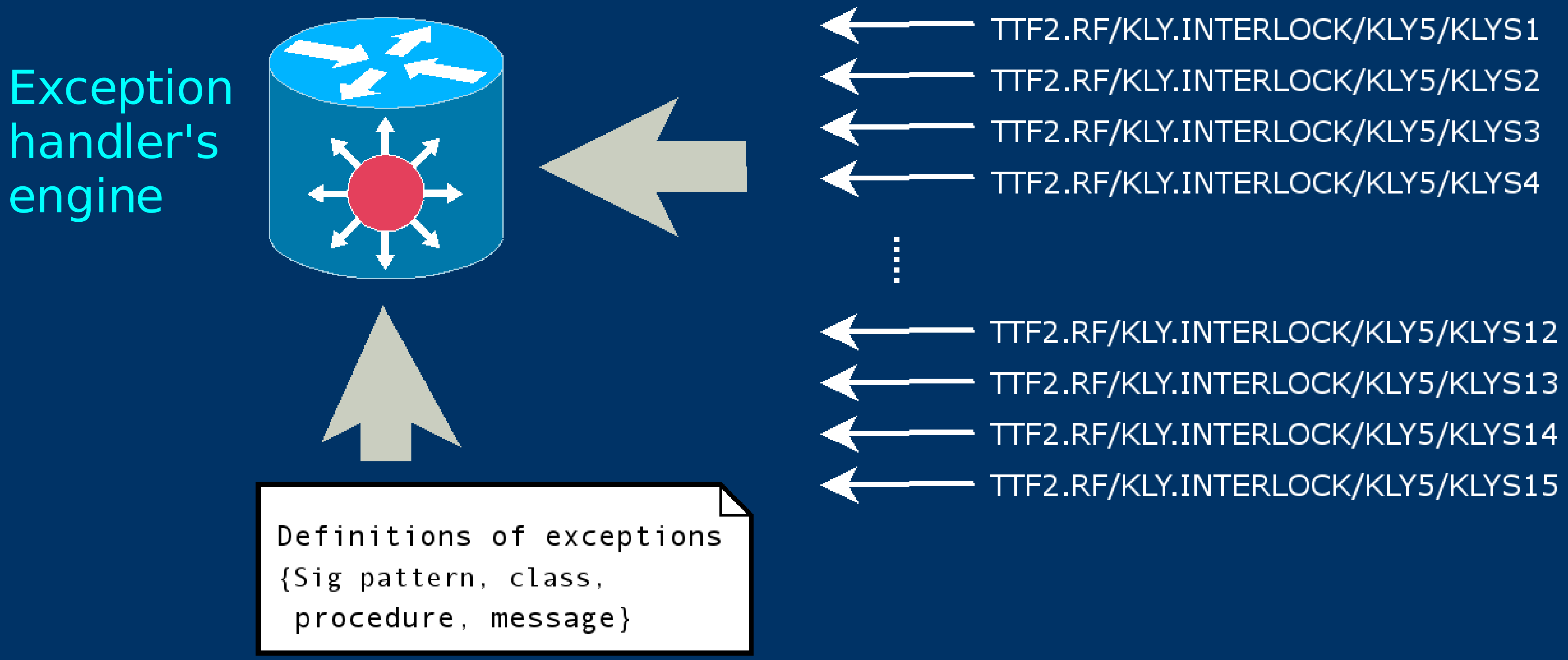
T=O(d^b)
S=O(d^b)

Now used:

- Depth first search with iterative deepening
  T=O(d),S=O(b^d)
- Cycles prevention
- Hill climbing

Role of exception handler execution engine:

- Discovering the fault patterns
- Conflict resolution based on faults categorization
- Execution of remedy procedure for encountered faults

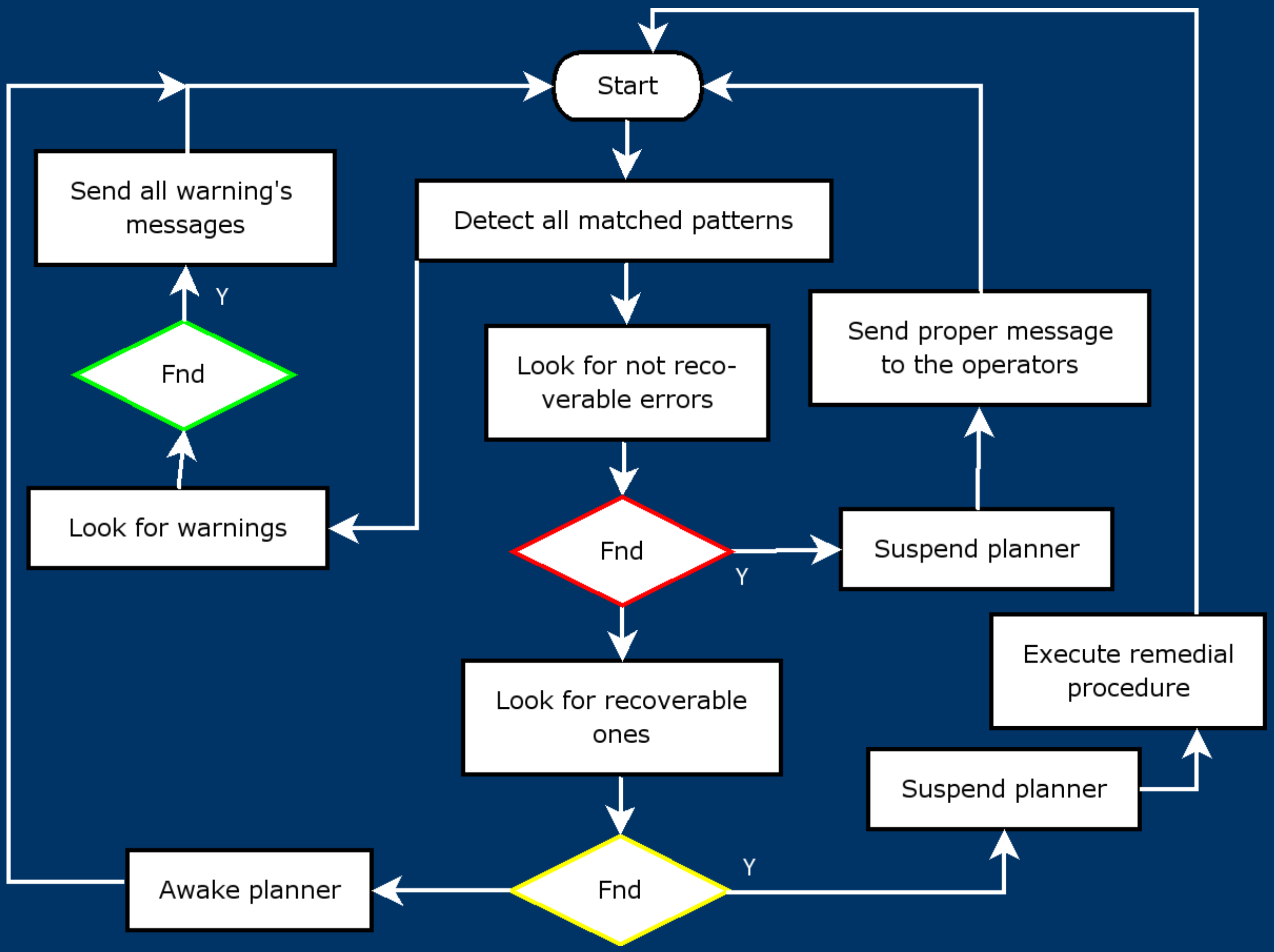


Pattern matching + conflict resolution

Conflict resolution:

- Fault arbitration based on
  - Categorization
  - Order of definition in specification file

Exception handler's policy:



Classes of faults:

- Warnings
- Recoverable
- Not recoverable

Cooperation scenarios

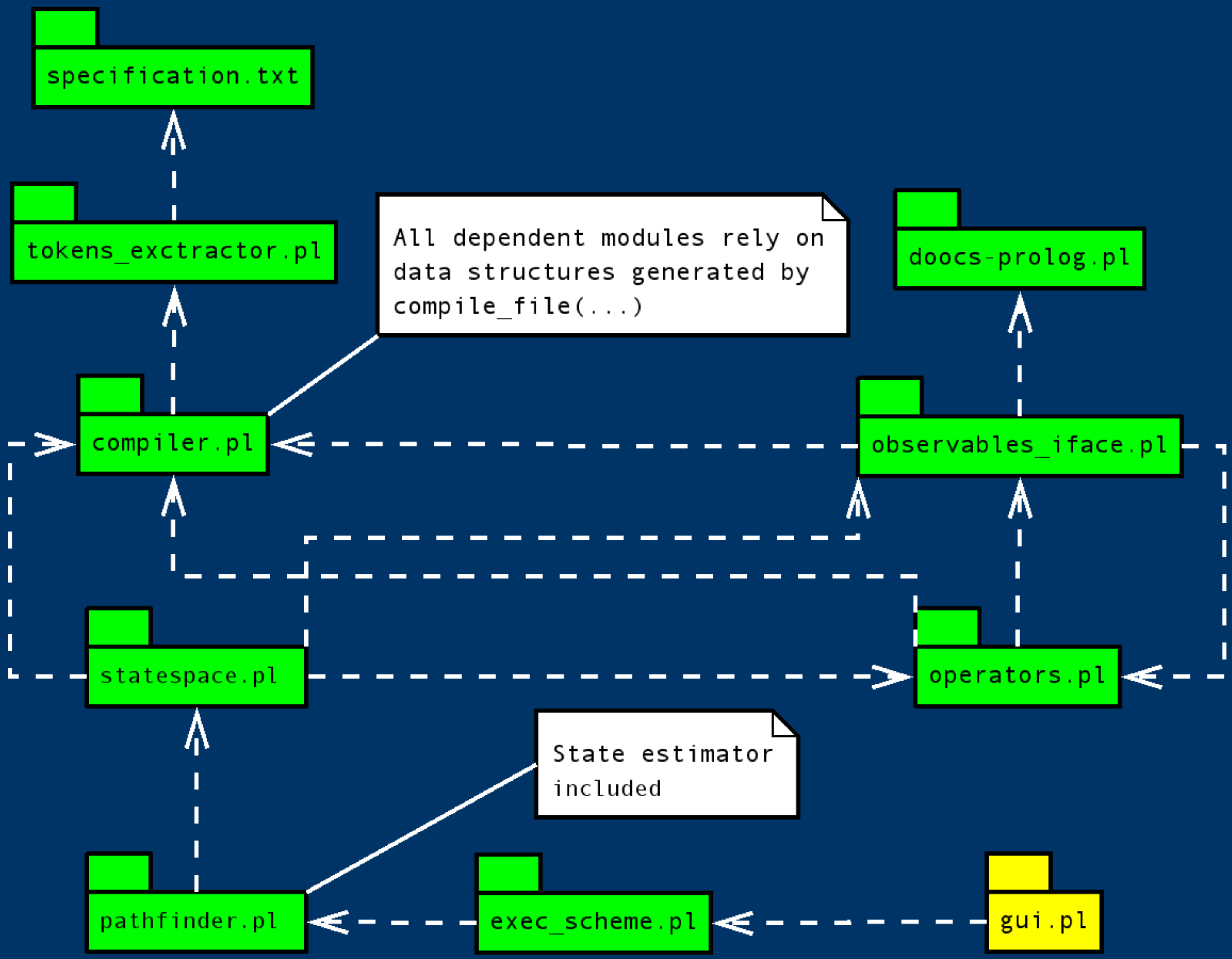
Scenario 1

- SE cannot estimate state
- indicate “incomplete”
- ask if EH can solve problem
- not
- remain in “incomplete”
- SE estimated state
- go back to automatic operation

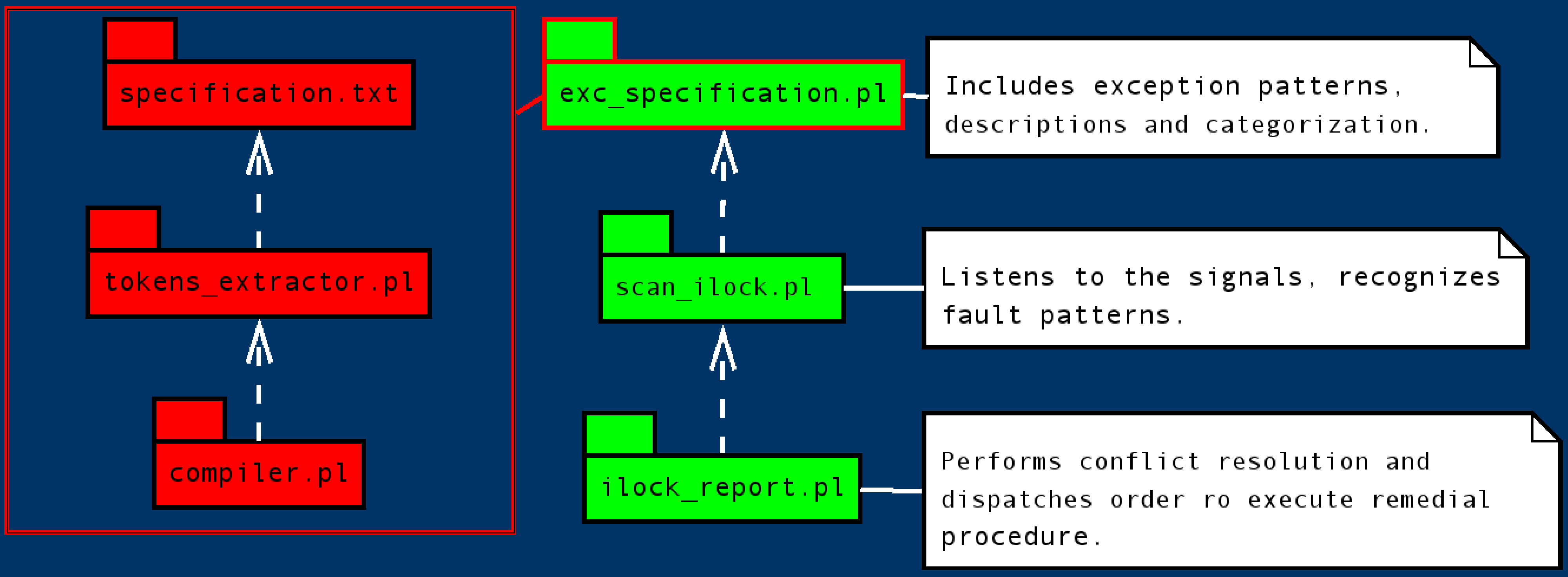
Scenario 2

- exceptions detected
- disable planner
- planner is suspended
- choose exception with the highest priority
- execute remedial procedure
- no exceptions
- awake planner
- planner back in operation

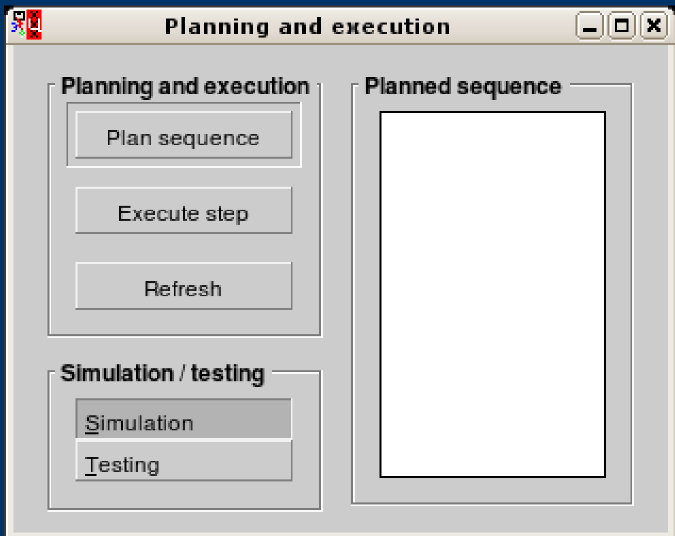
Implementation - planner

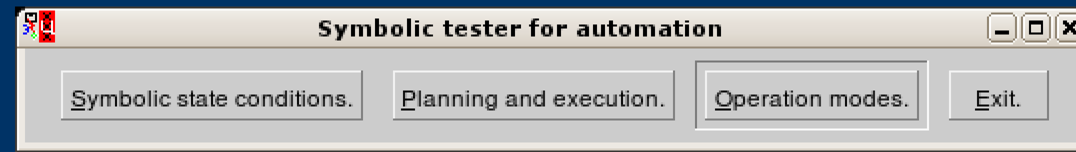


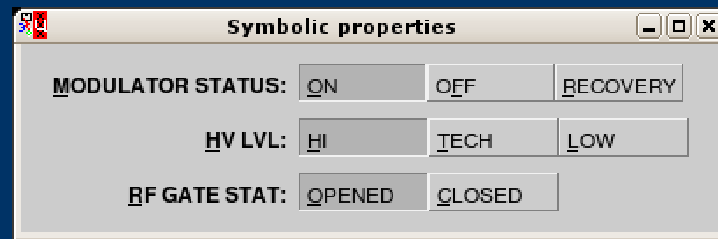
Implementation – exception handler

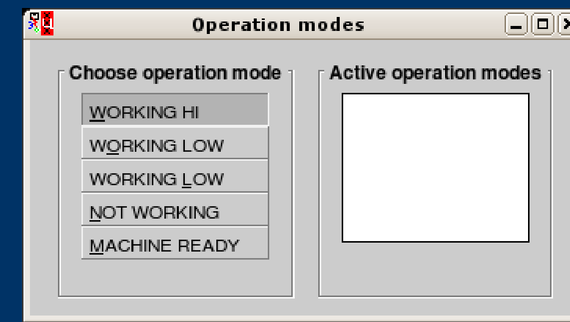


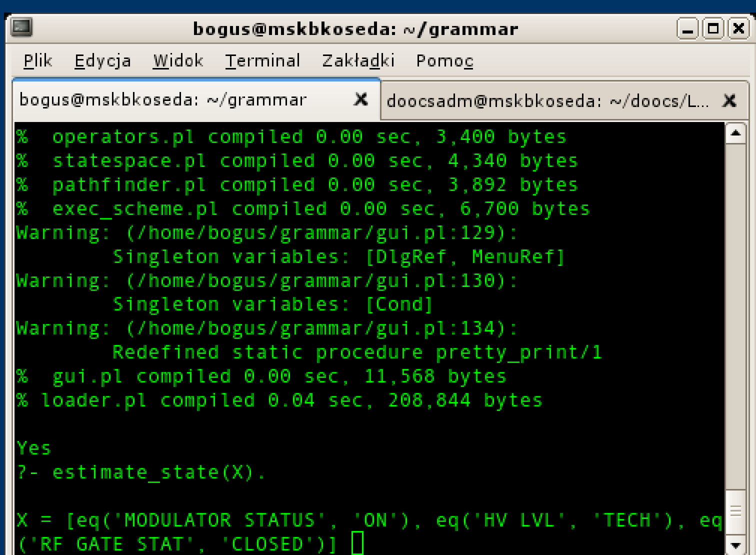
Twofold deployment

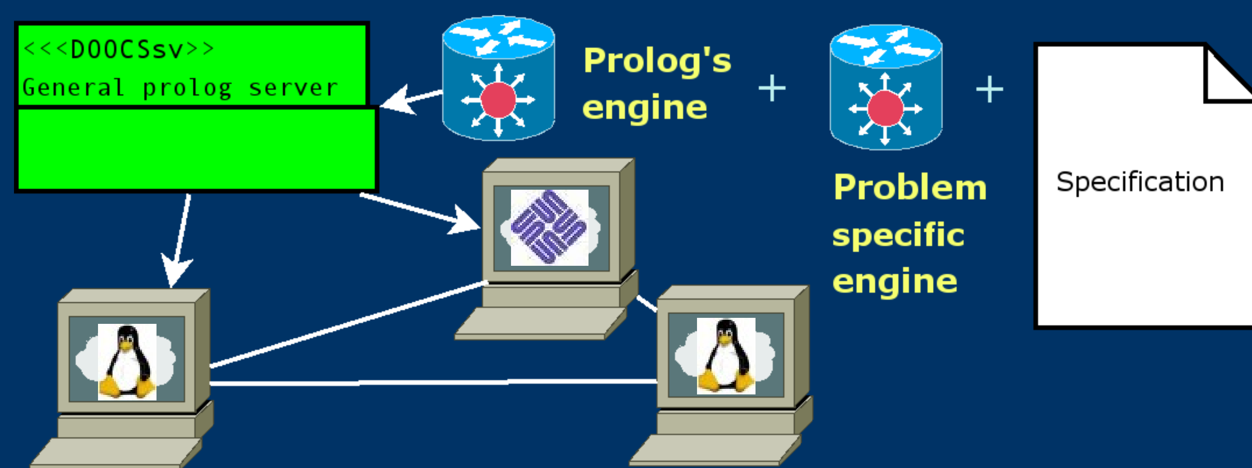












To be continued ...

Thank You