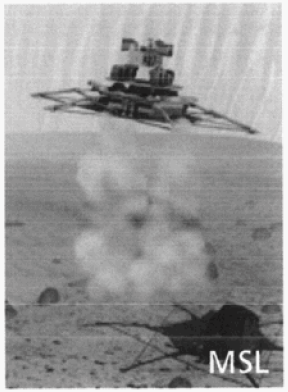
JPL

FINAL



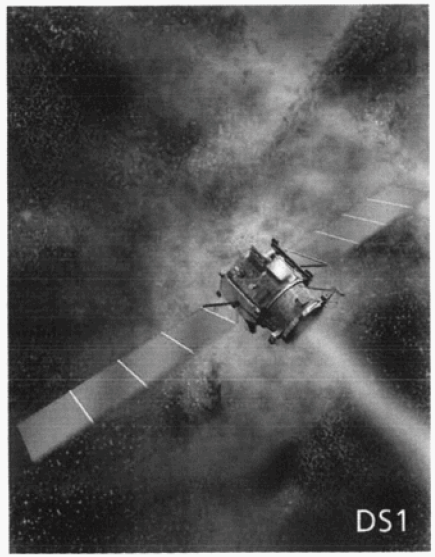
Prepared for the 2004
Software Assurance
Symposium (SAS)

Status report on:
Model Checking of
Artificial Intelligence
based Planners

Verifying AI Plan Models

Even the best laid plans need to be verified

Margaret Smith – PI
Gordon Cucullu
Gerard Holzmann
Benjamin Smith



Overview

- **Goal:** Using model checking, and specifically the SPIN model checker, retire a significant class of risks associated with the use of Artificial Intelligence (AI) Planners on Missions
  - Must provide tangible testing results to a mission using AI technology.
  - Should be possible to leverage the technique and tools throughout NASA.

- **FY04 Activities:**
  - Identify and select candidate risks
  - Develop and demonstrate technique for testing AI Planners/artifacts on:
    - A toy problem (imaging/downlinking) – demonstrate tangible results with an abstracted clock/timeline
    - A real problem (DS4/ST4 Champollion Mission) – demonstrate, using DS4 AI input models, that Spin can determine if an AI input model permits the AI planner to select 'bad plans'.

Identified Candidate Risks for Missions using Artificial Intelligence based Planners

- Mission Data Systems (MDS) was our target project when we submitted our proposal.
  - A large number of JPL AI community are working on the MDS project.
- Interviewed the MDS project personnel/JPL AI experts to discover risks.
- Ranked risks according to:
  - Feasibility:
    - Can the risk be addressed using model checking?
    - Are the necessary resources available?
  - Importance:
    - How concerned is the development team about the risk?
  - Commonality:
    - Can the results potentially be applied to other NASA AI planners or is the concern specific to MDS?
- These high ranking risks (close to 1, and circled in green on next two slides) will be addressed in our task:

current focus

- How do you know that an AI input model is consistent with only good plans and not with bad plans?
- How does the planner/scheduler react when two goals fail simultaneously?

Step 1: Identify Candidate Risks for Missions using Artificial Intelligence based Planners

Risk	Feasibility	Importance	Commonality	Rank
Each elaborator has its own thread of execution with the potential for race conditions.	5 - Appropriate application for model checking, but the absence of design documentation for MDS makes is difficult to derive Spin models.	3	5 - Goal elaborators are specific to the MDS Planner/Scheduler implementation.	**4**
Is the empty goal network safe? Is it possible to transition through unsafe configurations on the way to a 'safe' spacecraft state?	5 – Appropriate application for model checking, but the absence of design documentation for MDS makes is difficult to derive Spin models.	2	5 - Goal networks are specific to the MDS Planner/Scheduler implementation.	**4**
How does the planner/scheduler react when two goals fail simultaneously?	1 - Multiple goal failures can be modeled in Promela quite easily.	1	1 - This is a common concern for all AI planners.	**1**

Key:

- Risks we selected and will attempt to retire through this task

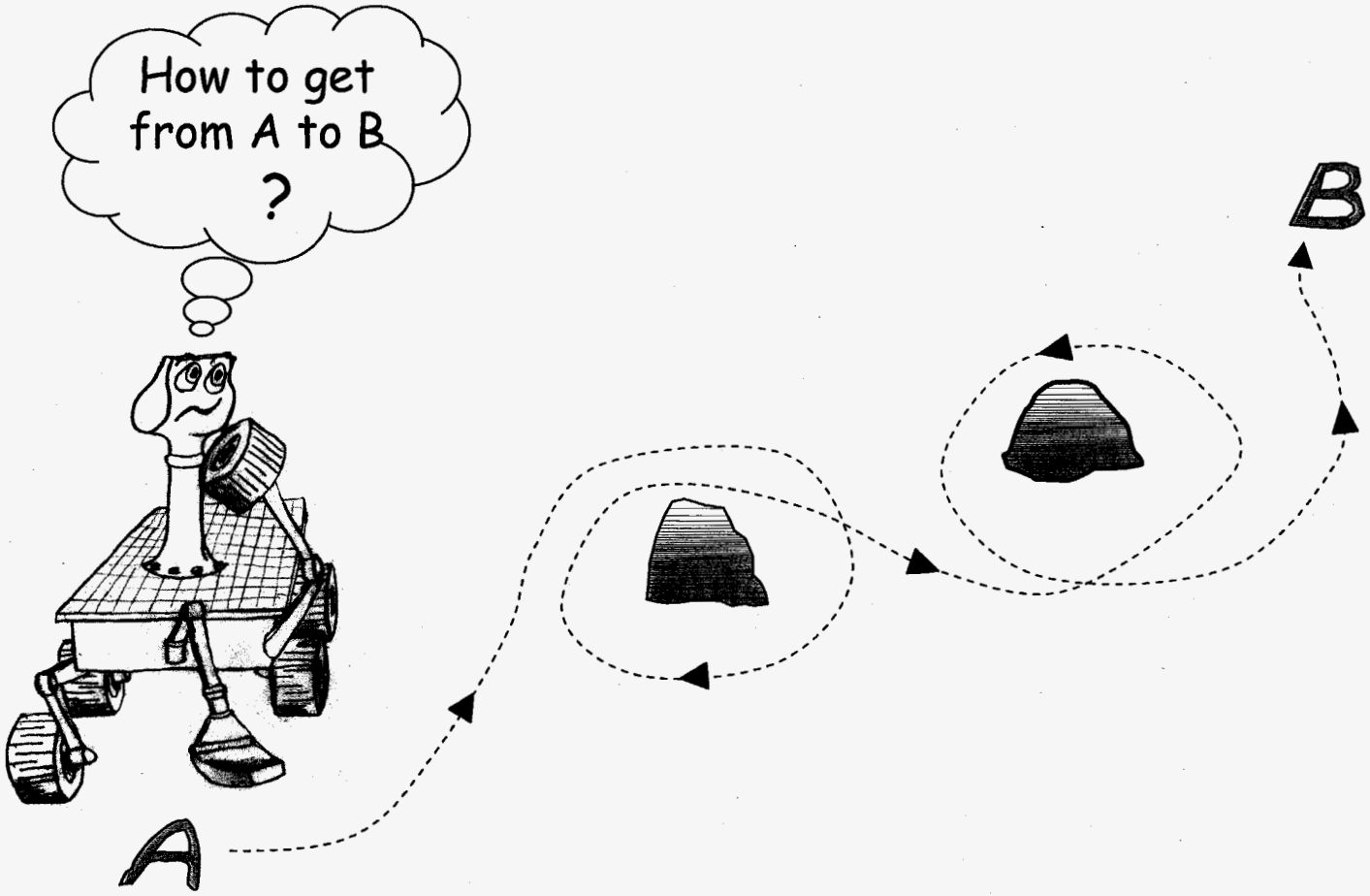
Candidate Risks - 2

Risk	Feasibility 1 – highest 5 - lowest	Importance 1 – highest 5 - lowest	Commonality 1 – highest 5 - lowest	Rank 1-best 5-worst
Does the MDS implementation meet it's requirements?	5 – Potential application for model checking, but the absence of design documentation for MDS will make it more difficult to derive Spin models.	3	5 – MDS Requirements and implementation are specific to the MDS Planner/Scheduler implementation.	**4**
If a plan exists will the planner find it in a reasonable amount of time?	5 – Spin analyses possibilities, and will not perform likelihood or performance analyses.	3	1 – This is a common concern for AI planners.	**3**
How do you know that an AI input model is consistent with only good plans and not with bad plans?	1 – AI input models can be expressed in Promela.	1	1 - This is a common concern for all AI planners.	**1**

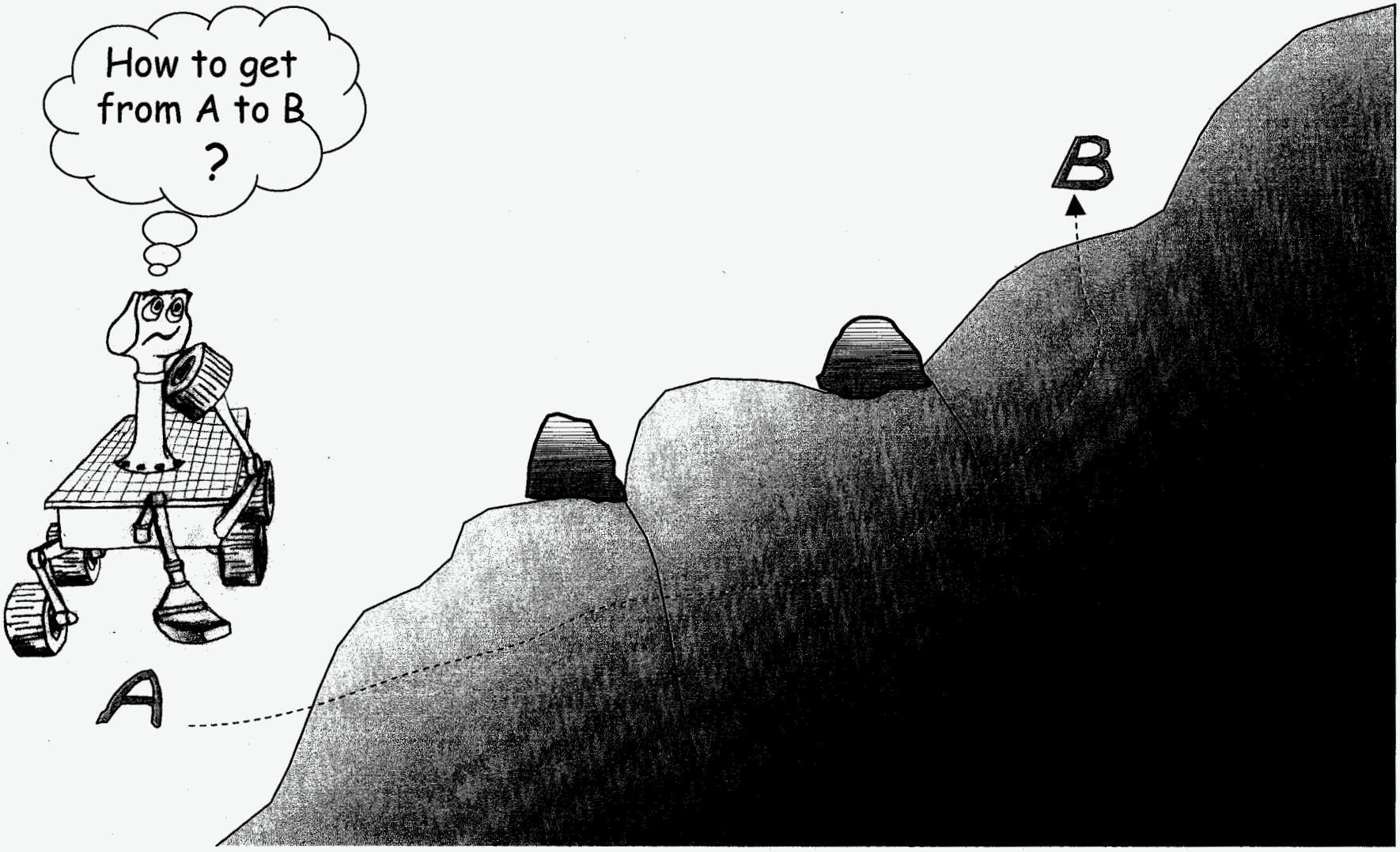
Key:

- Risks we selected and will attempt to retire through this task

Consequences of a bad plan
Wasted Resources



Consequences of a bad plan
Loss of Mission



Toy Problem: Imaging and Downlinking
Demonstration of

```
Activity Image {                        /*  image taking activity */
        dur = [10, 100];
        size = dur;
        use ssr size;             /* image has to be put in ssr memory */
};

Activity DL {                           /* downlink activity */
        dur = [100, 1000];
        vol = dur;
        use ssr -vol;             /* downlinking frees up memory */
        state DLWIN = open;  /* DL window has to be open */
};

state DLWIN = (open, closed);
resource ssr = [0,10000];

/* goals: */

Image 1 [5,10];    /*  start image between timepoint 5 and 10 – duration 100 */
Image 2 [100,110];     /*  start image between timepoints 100 and 110 – duration 100 */
image 3 [500,800];
image 4 [900,1000];

DL 1 [200,300]; /*  downlink window scheduled between  200 to 300 timepoints */
DL 2 [500,600];
DL 3 [800,900];
DL 4 [1100,1200];
```

Casper Model

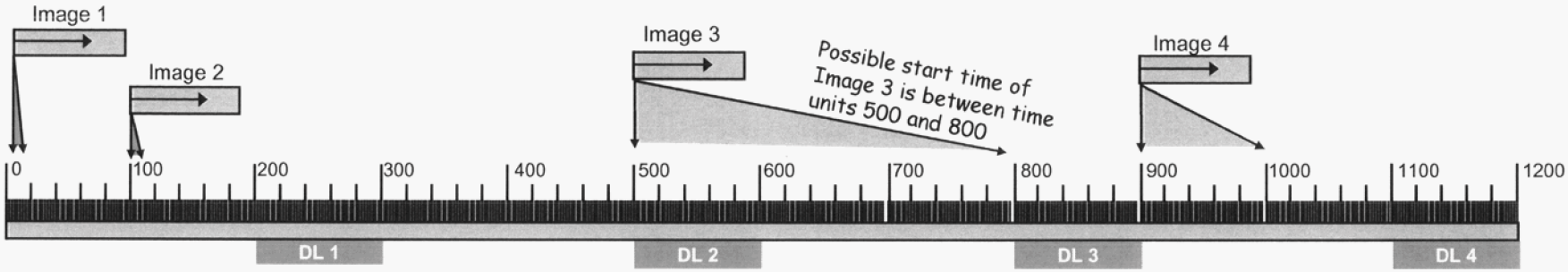
Goal: 4 images should be downloaded in 4 downlink windows

property: An image taken should eventually be downlinked

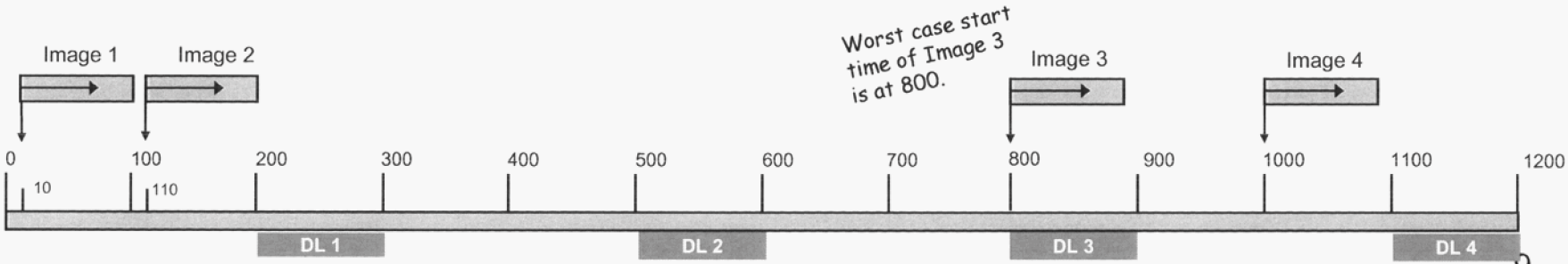
A desired/implied characteristic of imaging, but one that can not be expressed directly in the AI input model

Imaging and Downlinking model (continued-1)

Without abstraction: model clock explicitly, consider range of image lengths, consider range of image start times



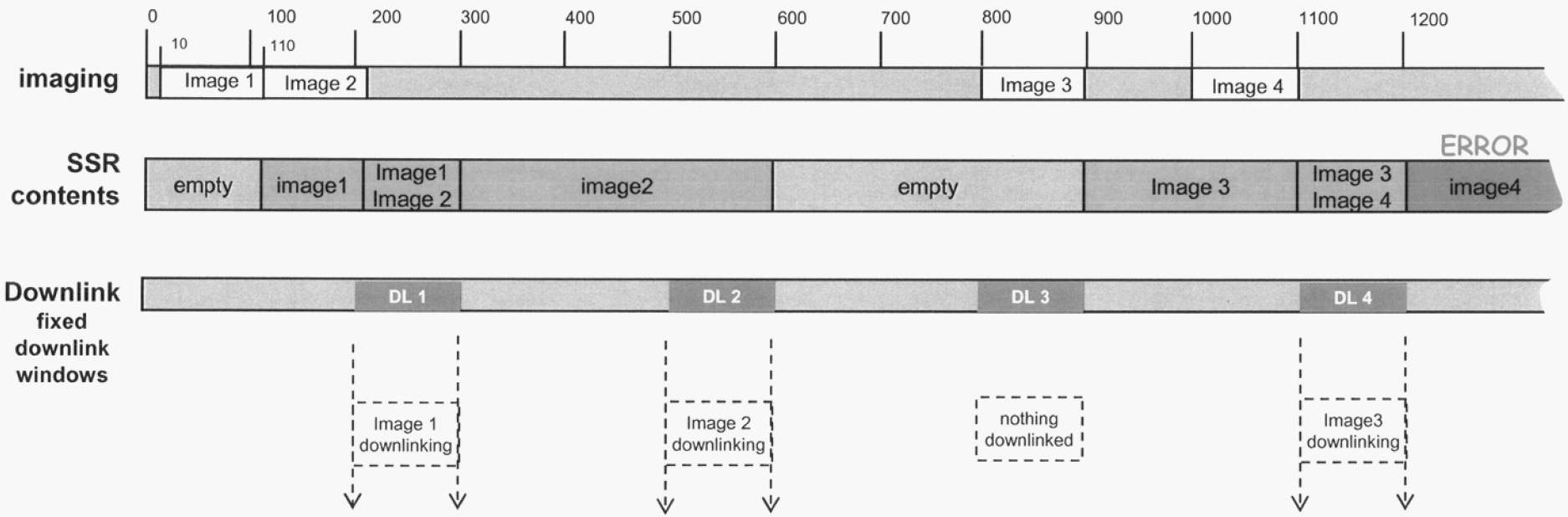
With abstraction: use time intervals instead of time points, consider worst case image lengths and worst case image start times



Imaging and Downlinking model (continued-1)

Error trace found by Spin model checker:

With this set of constraints it is possible for Image 4 to remain in the SSR at the end of the final downlink window



Checking DS4

A Real Problem

Deep Space 4 (DS4) /Champollion: A comet lander and sample return technology demonstration mission to Tempel 1 (cancelled).

⇨

Planned launch - 2003
Landed phase - 2006
Sample return - 2010

DS4 Requirements and a CASPER/ASPEN AI model are available

Goals for landed phase:

- Imaging
- Analysis of sub-surface samples involving:
  - Moving the drill to a 'hole'
  - Drilling
  - Mining for a sample
  - Moving the sample to an oven
  - Depositing the sample in an oven
  - Heating the sample and taking measurements

Challenge: check DS4 AI model to determine if a bad plan can be generated.

DS4 model elements

Goals:
3 Samples
2 Images

Activities:
Imaging
Drilling
Mining
Moving drill
Depositing sample
Oven experiment
Data compression
Data uplinking

Sample includes these activities

State variables:
oven1 & oven 2
(states: off-cool, on, off-warm, failed)
camera (states: off, on)
Drill location (states: hole 1, 3, or 7)

Resources:
2 ovens
1 camera
1 robotic arm with drill
Power (renewable)
Battery power (non-renewable)
Memory (non-renewable)

- *Goals* are satisfied by performing *Activities*.
- *Activities* are constrained by *Resource* availability and *State variables*
  Example: an oven must be in the 'off-cool' state in order to be selected for an oven experiment.
- *Activities* can change the values of *State variables* if no other activities have the lock and if the state transition is legal.
  Example: The oven experiment must be able to turn the oven to 'on'.

Defining good and bad plans

- A good plan contains all 5 memory using activities:
  - 3 samples
  - 2 images
- Therefore, a bad plan is a plan that does not contain 3 samples and 2 images
- Is it possible that this model permits bad plans?
  - How would the modeler test that the model can only produce good plans?

Standard Testing of an AI model

1. Construct the model from Science or other requirements.
2. Inspect the model for correctness against requirements.
3. Input the model to the AI planner and ask for a specified number of plans.
4. Manually inspect plans to identify bad plans

try again

bad plan(s)

Adjust constraints and other model elements to exclude bad plans.

all good plans(s)

End testing

Using Spin to exhaustively check for bad plans

- Each activity is represented as an independent Promela (the language of Spin) proctype
- All proctypes are instantiated in a non-divisible step.
- Activity/proctypes include their constraints for:
  - resource use and reservations
  - state variable values
  - other activities that must occur before, during or after activity in question.
- If a activity/proctype's constraints are met, the activity may proceed (be scheduled).
- In a Spin verification, all possible interleavings/schedulings are explored.
  - the timeline (clock) is abstracted to intervals or not included at all if possible
  - the assumption is that the scheduling window is long enough to accommodate all possible orderings of activities.

Representing AI model elements in Promela

Example CASPER/ASPEN model for taking a picture

Requests (goals)

```
take_picture take1 {
  start_time = 7h;
  file = "IMAGE1";
  no_permissions =
      ("delete");
};
take_picture take2 {
  start_time = 18h;
  file = "IMAGE1";
  no_permissions =
      ("delete");
};
```

Resources

```
Resource civa {// camera
  type = atomic;
};
Resource comm {
  type = atomic;
};
Resource data_buffer {
  type = depletable;
  capacity = 30;
  min_value = 0;
};
```

Activity

```
Activity take_picture {
  RawImageSize rwis1;
  string file;
  start_time = [10, infinity];
  duration = [1m,10m];
  reservations =
    comm, data_buffer use 5,
    civa,
    civa_sv must_be "on";
}
```

States

```
State_variable civa_sv {
  states = ("on", "off",
      "failed");
  default_state = "off";
};
```

Representing AI model elements in Promela

Example Promela model for taking a picture

```
unsigned data_buffer : 3 = 4;
mtype = { on, off, failed, ... };
bool civa = 1; /* atomic resource: 1 is available, 0 is in use */
unsigned count : 3; /* # of memory using activities scheduled */
mtype civa_sv = off;
chan mutex_civa = [2] of {pid}; /* queue for reservations */
```

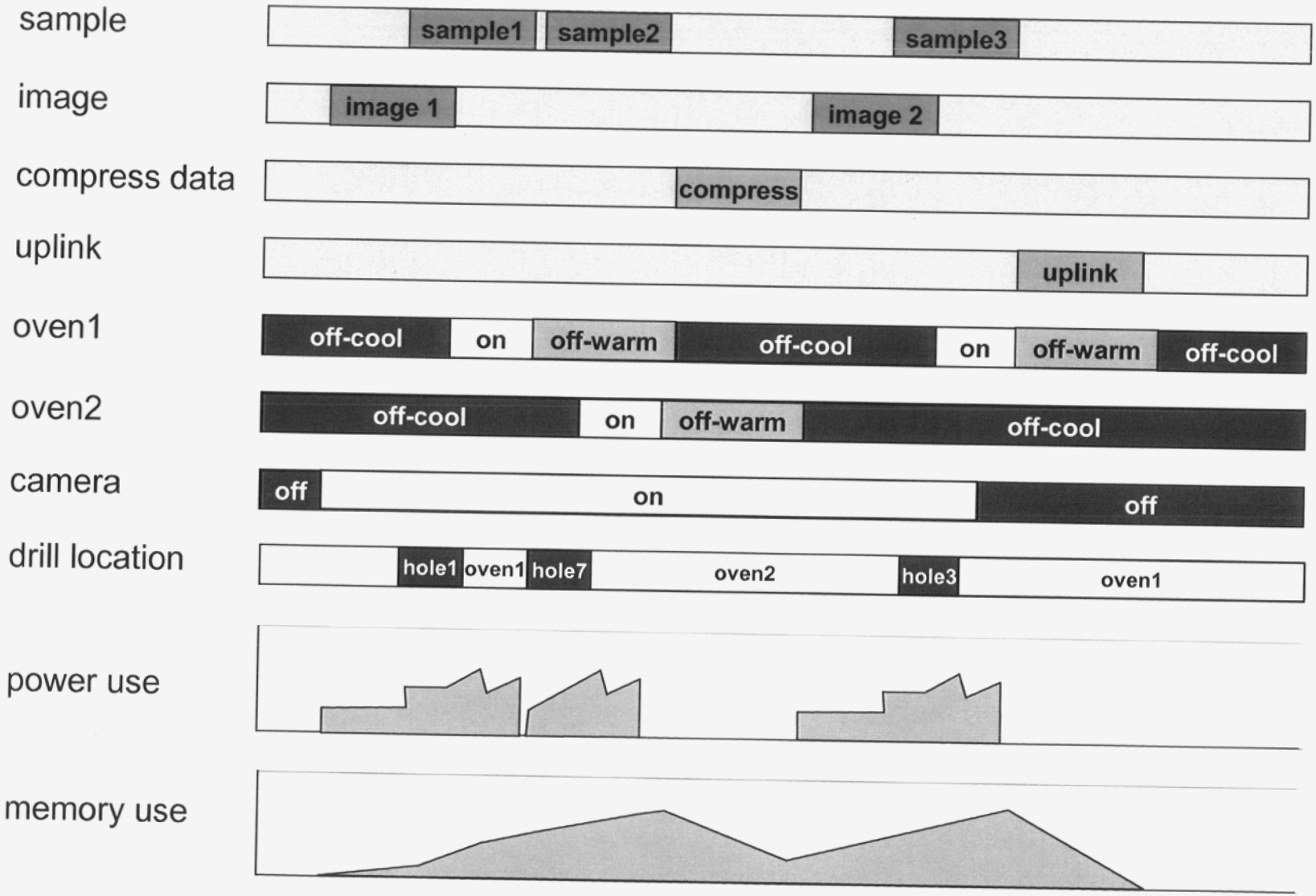
```
proctype take_picture() {
  /* civa_sv must be on and civa must be available */
  atomic { (((civa_sv == on) || empty(mutex_civa)) \
  && civa && ((data_buffer - 1) >= 0)) ->
    if :: (civa_sv != on) -> civa_sv = on :: else fi;
    mutex_civa!_pid; /* 'must_be' so reserve civa var */
    data_buffer = data_buffer - 1;
    civa = 0; /* camera in use */
    plan!picture; /* take picture */
    count = count + 1; /* variable needed for property * }

    d_step {
      civa = 1; /* picture complete - give back camera */
      mutex_civa??eval(_pid);    }
}
```

```
Init {
      atomic {
            ...
            run take_picture();
            run take_picture();
            ...      }
}
```

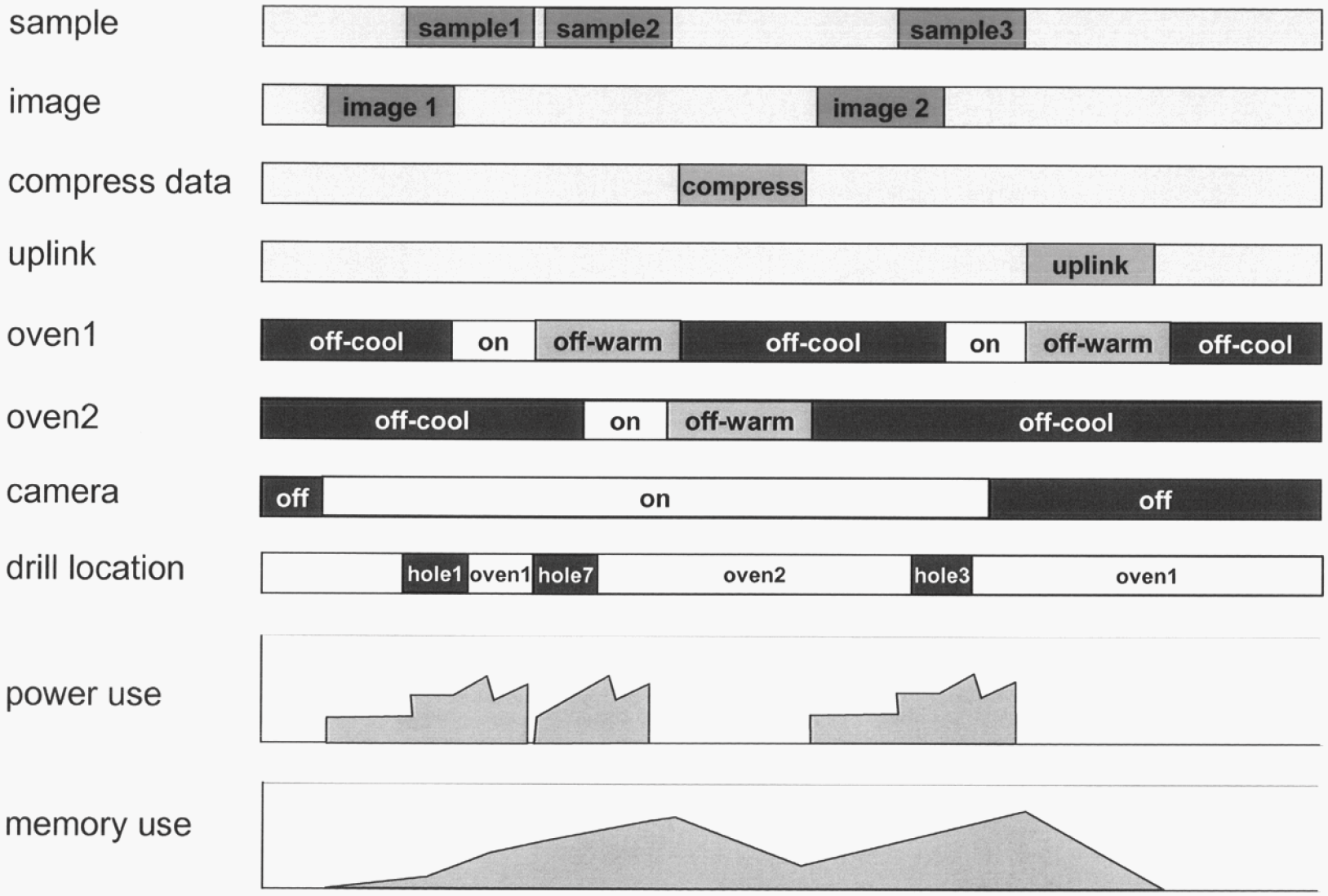
A good plan for DS4

is when all goals (in green) are met



A plan

is when all goals (in **green**) are met



Property for exposing bad plans

All plans must include all five goals (3 samples, 2 images)

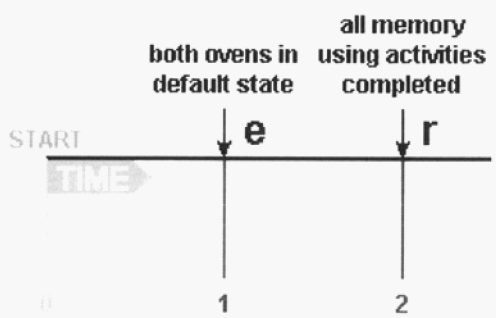
Never Claim for C:/mhs/proj/MAP/AI_models/models/ds4/promela/ds4_un...

```
#define p1          (count == 5)   /* all memory using activities completed */
#define p2          ((oven1_sv == off_cool) && (oven2_sv == off_cool))   /* both ovens in default state *
/
never {
S0:         do
            :: p2 -> goto acceptS1
            :: true
            od;
acceptS1:
            do
            :: p1 -> goto S2
            :: !p1
            od;
S2:         0           /*full compliance if reached*/
}
```

Close

Timeline Editor - C:/mhs/proj/MAP/AI_mo...

File Tools

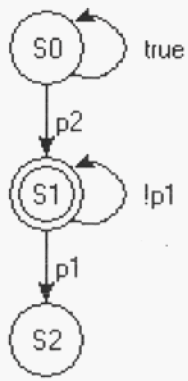


Graphical Automaton for C:/mhs/proj/MAP/AI_models/model...

KEY:

p1 all memory using activities completed

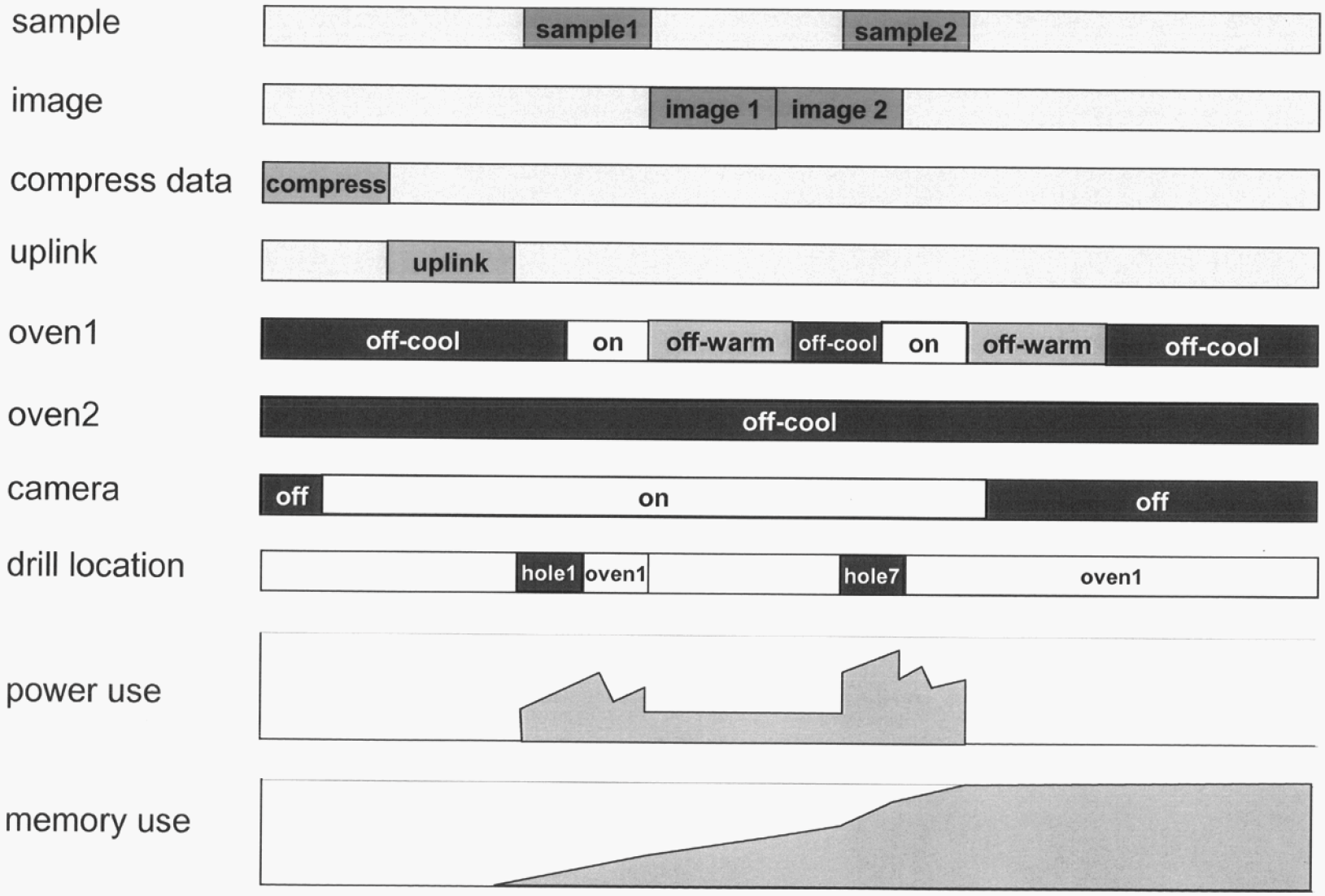
p2 both ovens in default state



See Text | Dismiss

A bad plan found by Spin

only 4 goals (in green) are met



Fix constraints and recheck

- Added a constraint to the AI model that 'compression' may only be performed if the data buffer is non-empty

- Rechecked property using Spin
  - an exhaustive check shows that all plans contain the five goals.

AI Model Testing Process using Spin

- Construct the model from Science or other requirements.
- Inspect the model for correctness against requirements.
- Formulate 'good plan' properties
- Express model in Promela and **exhaustively check** using Spin.

Replaces sampling

Replaces manual inspection of samples

try again

bad plan (error trace)

Adjust constraints and other model elements to exclude bad plans.

no errors

End testing

Next Steps

- Working with the former DS4/ST4 development team to discover additional properties types that we can check.
- Will explore the possibility of automated conversion from Promela models to CASPER/ASPEN models.
- Will explore a applying this technique to a project that is actively using CASPER/ASPEN:
  - 3 Corner Sat
  - Earth Orbiter 1

Backup

CASPER / ASPEN

ASPEN: Automated Scheduling and Planning Environment

A modular, reconfigurable application framework, capable of supporting a wide variety of planning and scheduling applications, that includes:

- an expressive modeling language
- a resource management system
- a temporal reasoning system
- and a graphical interface

CASPER: Continuous Activity Scheduling Planning Execution and Re-planning

- Supports *continuous* modification and updating of a current working plan in light of changing operating context
- Applications:
  - Autonomous Spacecraft – 3CS
  - Autonomous Spacecraft – TS-21
  - Rover Sequence Generation
  - Distributed Rovers
  - CLEar (Closed Loop Execution and Recovery)