UNIVERSITÄT KLAGENFURT

Why even old AI-folks get very excited about Answer Set Programming

Gerhard Friedrich

Institute for Applied Informatics
Alpen-Adria Universität @ Klagenfurt.AUSTRIA

Congratulations …

… to the Italian Association for LP for your 25th anniversary.

Italy has a vivid LP community with many bright researchers.

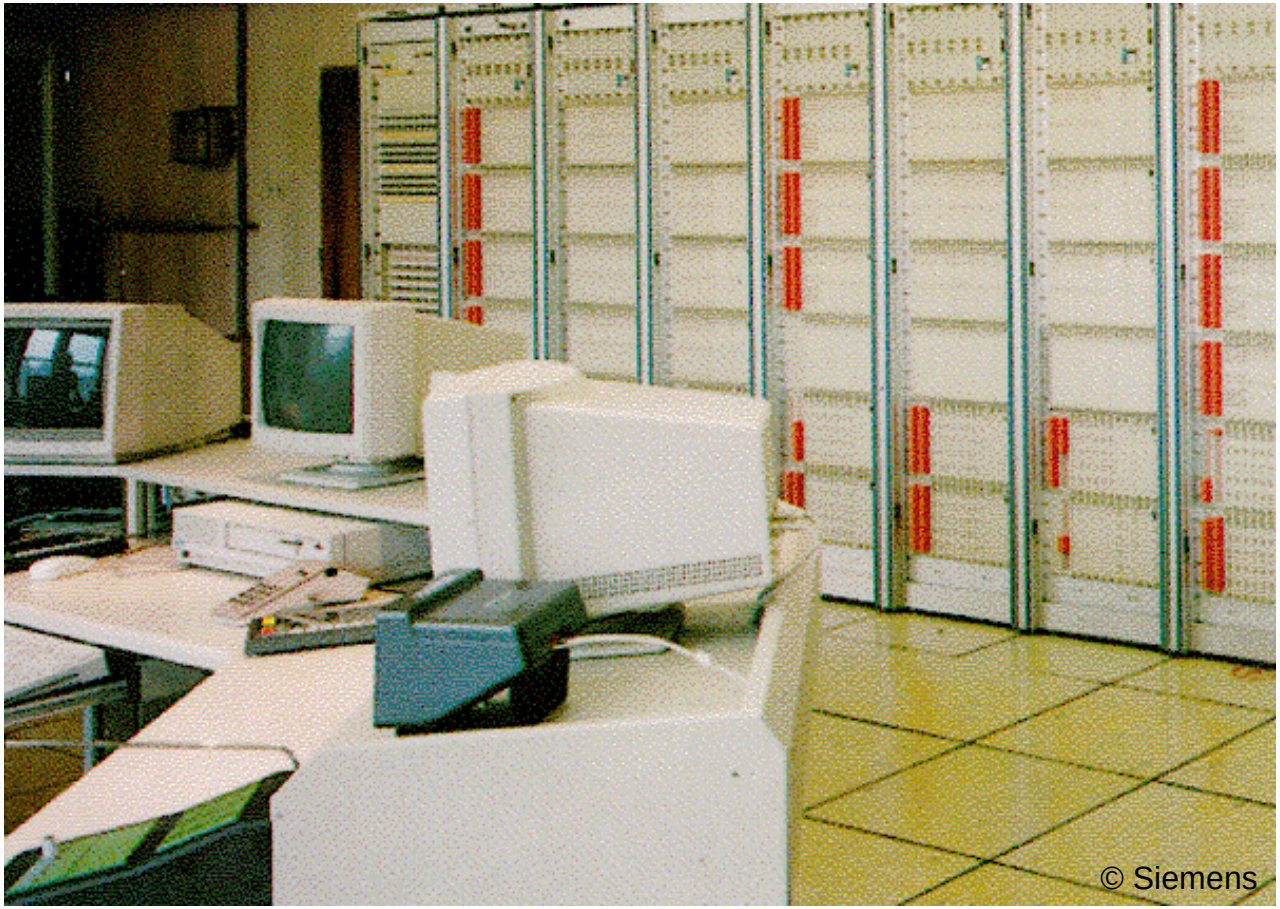
My presents:

- Yes, you are on track, please give us more
- A gift box of problems which deserve solutions from the viewpoints of industry and AI sub-communities

Status 1987 – 1991

Task: Develop a configuration and diagnosis system.

Subsequently we tried Datalog and Description Logic.



Status 2010

Special tracks in LP conferences on applications of ASP, e.g.

- planning
- diagnosis
- configuration

“old” AI-folks get really excited,
you can see the wow-effect.

Mission completed?
Declare victory?

Structure

- Repair
  - ASP successfully applied
- Diagnosis
  - Putting things together is the great challenge
- Configuration
  - Dispelling some common myths

Self-healing Service-Based Processes

Joint work with:
Mariagrazia Fugini, Politecnico di Milano
Enrico Mussi, Politecnico di Milano
Barbara Pernici, Politecnico di Milano
Gaston Tagni, Vrije Universiteit Amsterdam

Developed within the FET project WS-Diamond (Web Service Diagnosis, Monitoring, and Diagnosability) funded by the EU.
See http://wsdiamond.di.unito.it for further information.

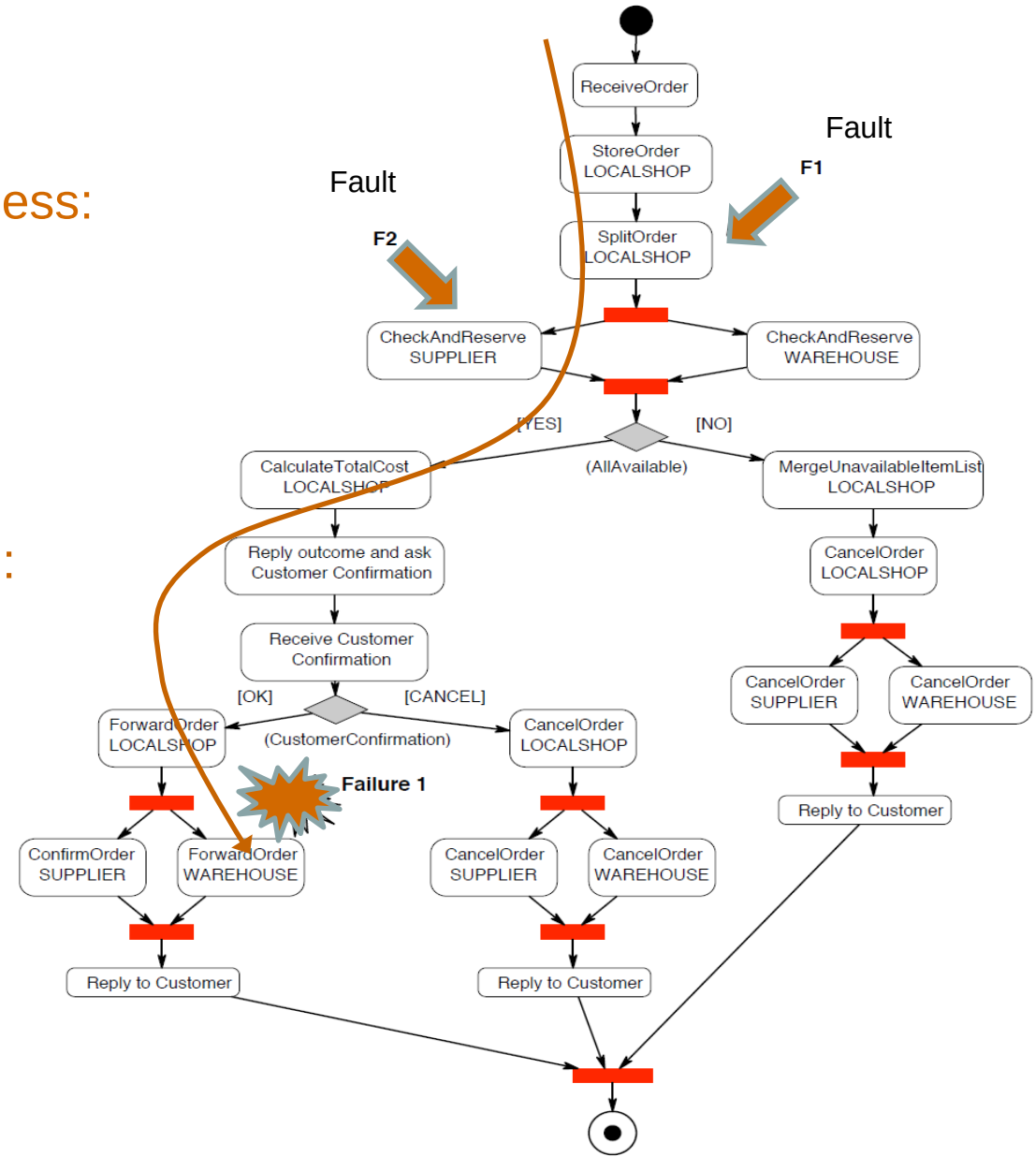
Wanted: general method

Given:

Description of (distributed) process:
- Control flow
- I/O-dependencies of activities but **not** the semantics
- Repair actions

Description of process instance:
- Execution log
- Failure observations

Questions:
- Can the faulty instance be completed correctly?
- What is the most cost efficient repair?



Basically two approaches to deal with failures

1. Software engineers try to anticipate all failures and design an exception handling mechanism

Problem: Human effort,
correctness, completeness, cost efficient repairs?

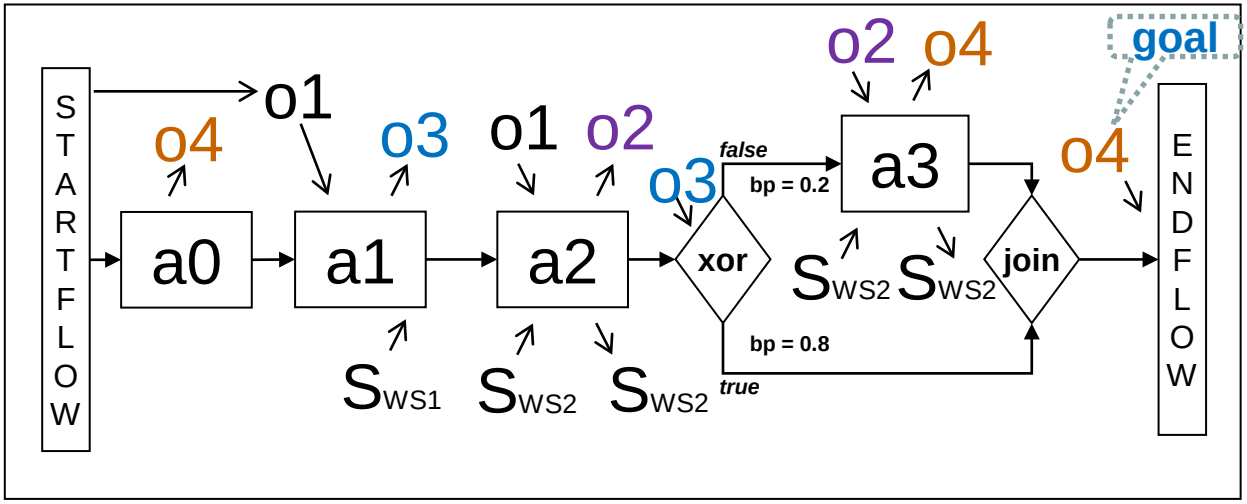
2. Exploit the available information and semantics (model-based) and generate repair plans for observed failure situations

Advantage: Correct repairs (under some assumptions),
explore the search space exhaustively,
save human effort

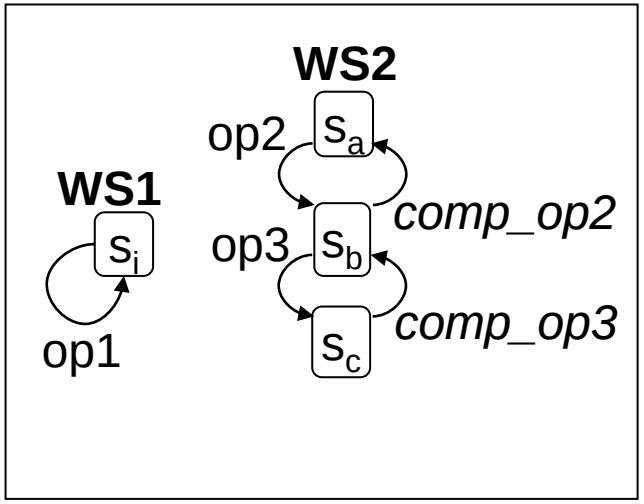
Problem: Computational costs

Modelling processes for repair

Process (schema)



Conversation models

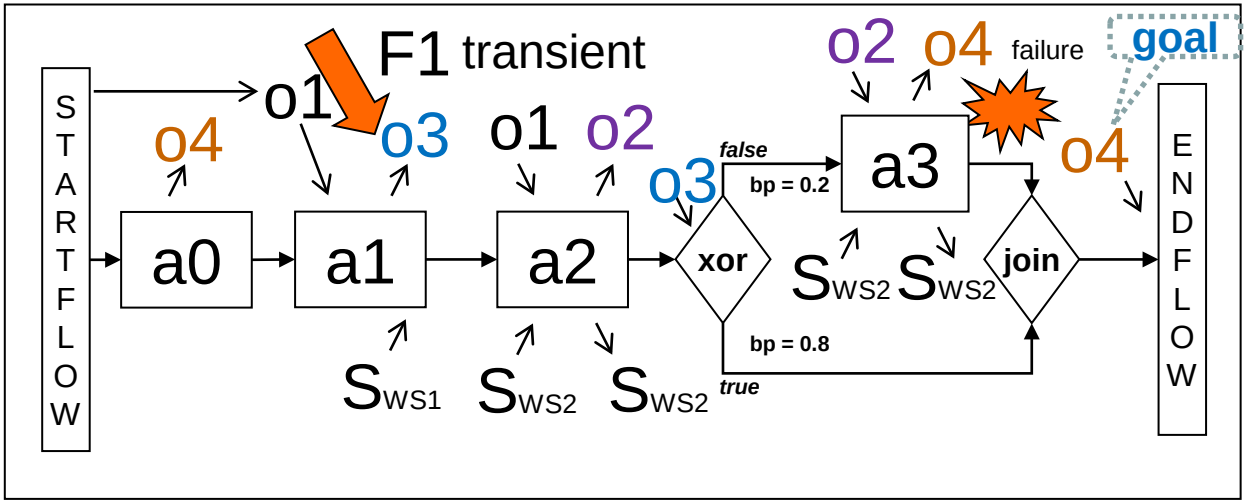


Process instance is set of states of
- activities
- control edges
- objects

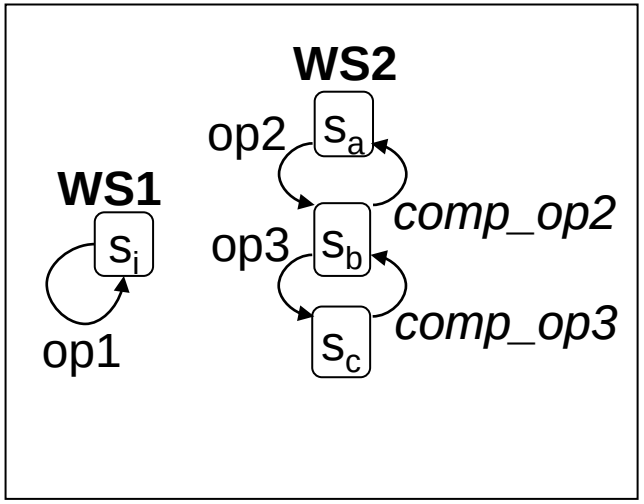
Process execution is sequence of instances: $(I_0, \ldots, I_n)$

Modelling processes for repair

Process (schema)



Conversation models



Faults:

- Non intended activity behaviour
- permanent/transient

Repair actions:

- retry
- compensate, one-step-back
- substitute

Correctness of object states

$ObjectState^{I}(o)$ … object state of object o in instance I

Given a process S,
an initial process instance I_0 of S,
a process instance I, and
a set of objects Obj:

The states of objects in Obj are correct in I *iff*
there exists a process execution without faults $(I_0, \ldots, I_n)$ of S s.t.
for all $o \in Obj$: $ObjectState^{I}(o) = ObjectState^{I_n}(o)$.

Repair of faulty process instance

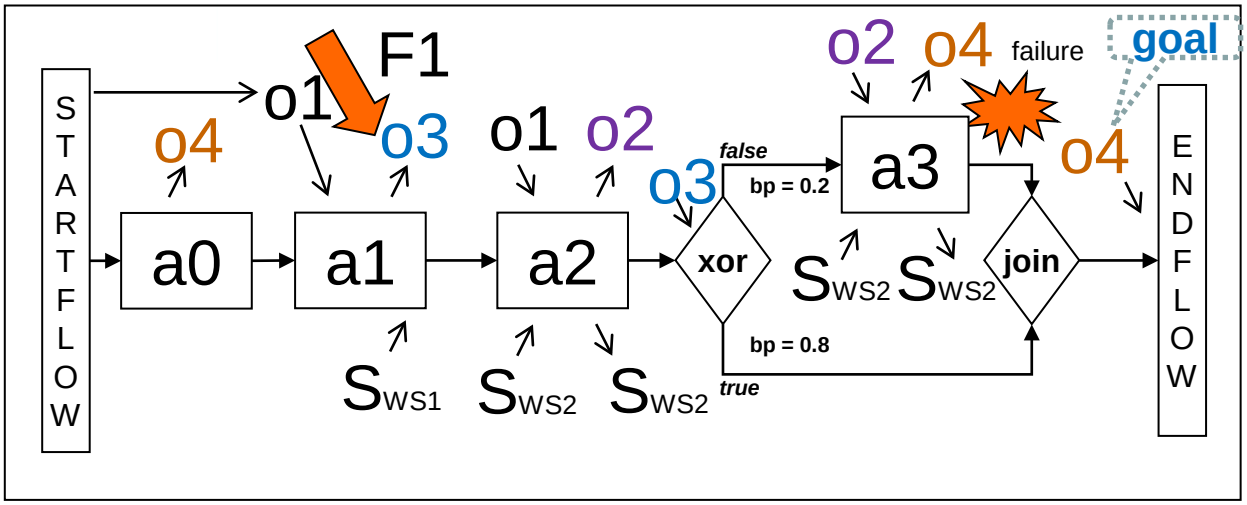
Given a process **S**,
an initial instance $\mathbf{I_0}$ of **S**,
a faulty instance $\mathbf{I_f}$,
the set of goal objects **GO**,
a set of repair actions **RA**,
and a sequence of repair actions **R**, all of which belong to **RA**

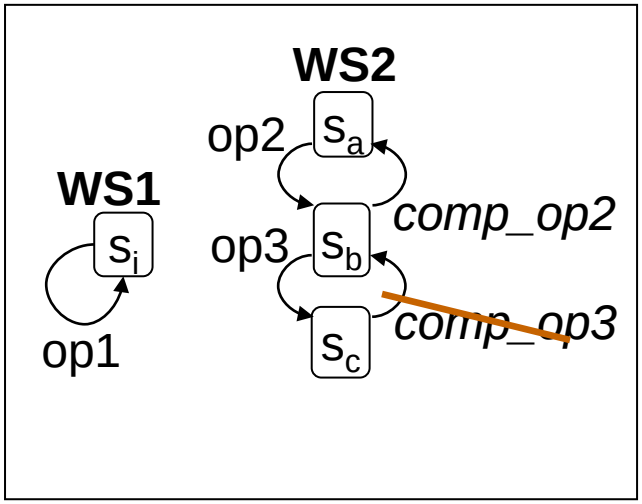
R is a **repair** for $\mathbf{I_f}$ iff

the set of object states of the goal objects **GO** in the repaired instance $\mathbf{I_R}$
produced by applying **R** on faulty instance $\mathbf{I_f}$

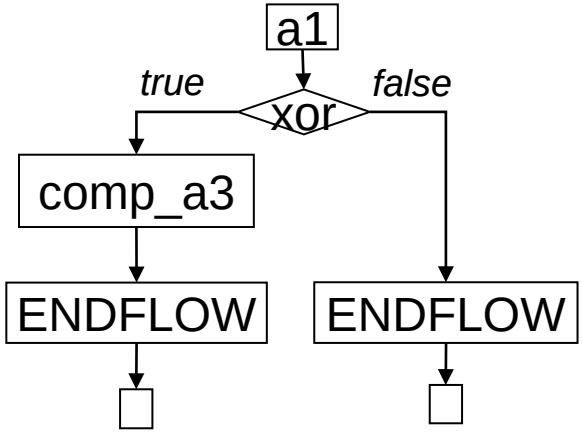
are the same as in a correct execution from $\mathbf{I_0}$ to **ENDFLOW** without faults.

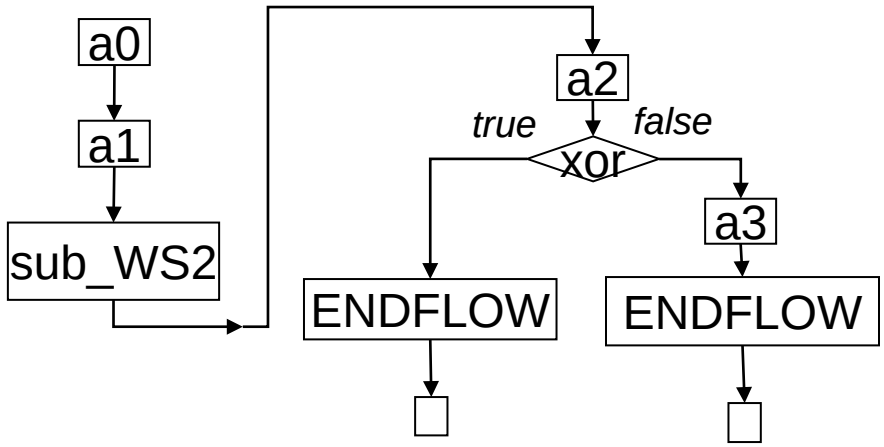
Repair at run time





Successful repair plans: conditional, forward branching time structure





Computing successful repair plans

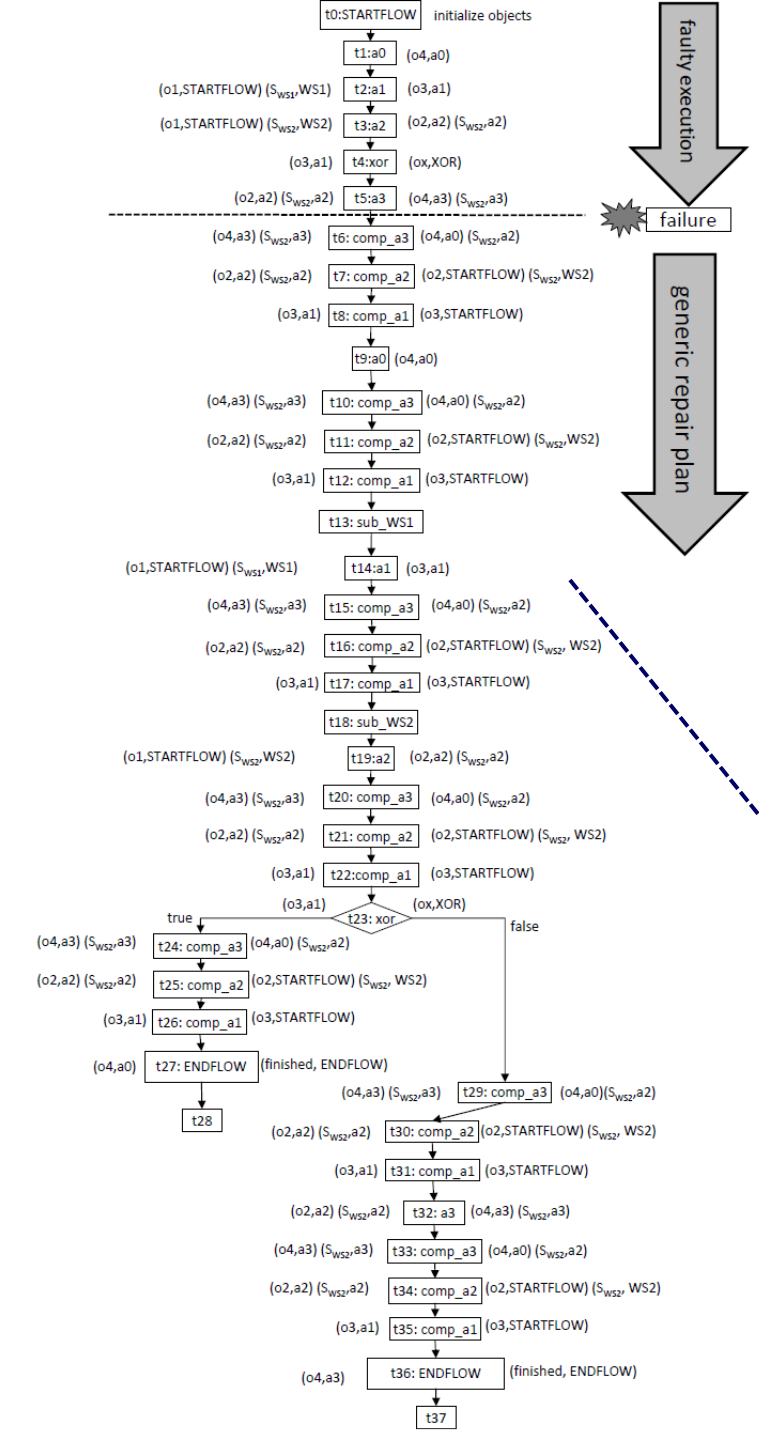
Given n repair actions, there exist at least $o(n^n)$ possible repair plans

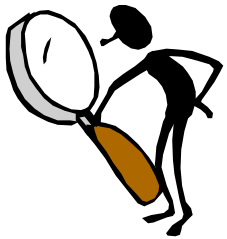
- Apply heuristics to limit the search space

Strategy:

- In order to generate correct states of goal objects at least a subsequent of a path of the process must be executed
- Between the (re-)execution of activities compensation and substitution actions may be added in order to allow the activity to produce correct outputs
- Since we do not know which repair actions must be executed we generate a generic repair plan
- Repair plans are generated from the generic repair plan by assigning to each action either apply or do not apply

Generic repair plan





Logical representation

LOG contains:

Execution

```
do(STARTFLOW,t0), do(a0,t1), do(a1,t2), ...
```

Diagnosis

```
ok(STARTFLOW,t0), ok(a0,t1), transientFault(a1,t2), ...
```

Time structure

```
next(t0,t1), next(t1,t2), next(t2,t3), ...
```

Plan:

```
do(a1,t6), do(xor,t7), followOn(t7,t8,true),
   do(comp_a3,t8), ...
```

Logical representation

S ... logical description of the process structure, available repair actions, ... (i.e. application specific data)

LOG ... logical description of repair case

R ... logical description of the repair plan

(generic repair plan GR + information of action application)

RP ... logical description of the repair problem

S, LOG, R, RP are mutually consistent.

RP is a logical theory describing:

- Correctness of object states at time points: `correct(o,s,t)`.
  Each object has at each time point exactly one state
- Preconditions and effects of actions
- A distinct object `finished` is introduced
  written by ENDFLOW depending on all goal objects
  (in order to avoid reasoning about sets of object states)

Implementation

Apply disjunctive logic programming (DLV)

Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, Francesco Scarcello: The DLV system for knowledge representation and reasoning. ACM Trans. Comput. Log. (2006)

- Model the application of actions `a` at time points `t` as disjunction:

  $\texttt{do(a,t)} \vee \texttt{skip(a,t)} \leftarrow \texttt{precondition.}$

- Each logical model contains a repair plan
- Prune logical models by

  $\texttt{final\_timepoint}(T_e) \wedge$
  $\texttt{state\_of\_at(State, finished,}\ T_e) \wedge$
  $\texttt{not correct(finished, State,}\ T_e) \rightarrow \texttt{false.}$

DLV outputs minimal (stable) logical models.
These models contain successful repair plans.

Search method for cost minimal repairs for free – soft constrains.

Completeness

1) Rewrite processes to Single Assignment Form:

Objects are written once, read multiple times

2) Compensations of service states and process objects can be executed independently without precondition

It follows: repair actions do not interfere,
no correct object states are destroyed.

Under these conditions completeness can be guaranteed.
(i.e. a successful repair plan is found if there is one)

Are there other practically relevant characterisations?

Summary of ASP and Repair

Approach for self-healing of processes and implementation
(such as orchestrated web-services, workflows, ...)

- Disjunctive logic programming applied to formalize and compute:

  The ideas of the DLV-Action language were exploited.
  This was extremely helpful, thank you.

- Approach is model-based,
  i.e. the generation of diagnoses and repairs is based on the model of the process and the repair capabilities.
  - No need to “program” fault recovery procedures
  - Correctness provided by formalizing the semantics of actions
  - Completeness can be provided for classes of problems

Untold stories and open questions

No complete diagnosis and repair method available for the general case:

Input is a set of “faults”, enhance to set of “leading” diagnoses.

No limits on preconditions and effects of actions.

It’s all about heuristics: We were only successful because of the “generic repair plan” heuristic.

Assistance needed, please continue:

“real-time” search: The more time available, the better the result.

Control over the model-generation process would be appreciated. (Note: non-minimal models can be acceptable solutions)

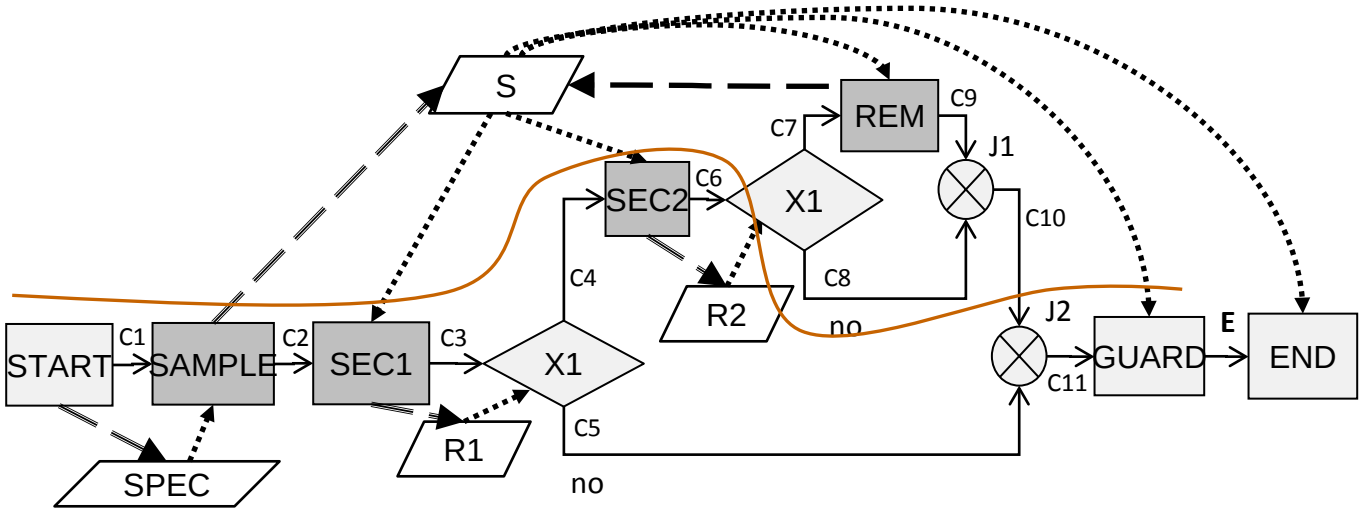
Existential quantification: Executing an action implies the existence of a successor state.

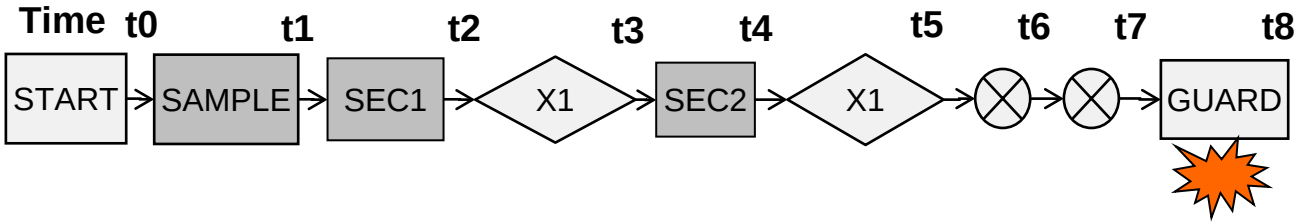
Structure

- Repair
- Diagnosis
- Configuration

Diagnosis of process trajectories …

…under partially known behavior.



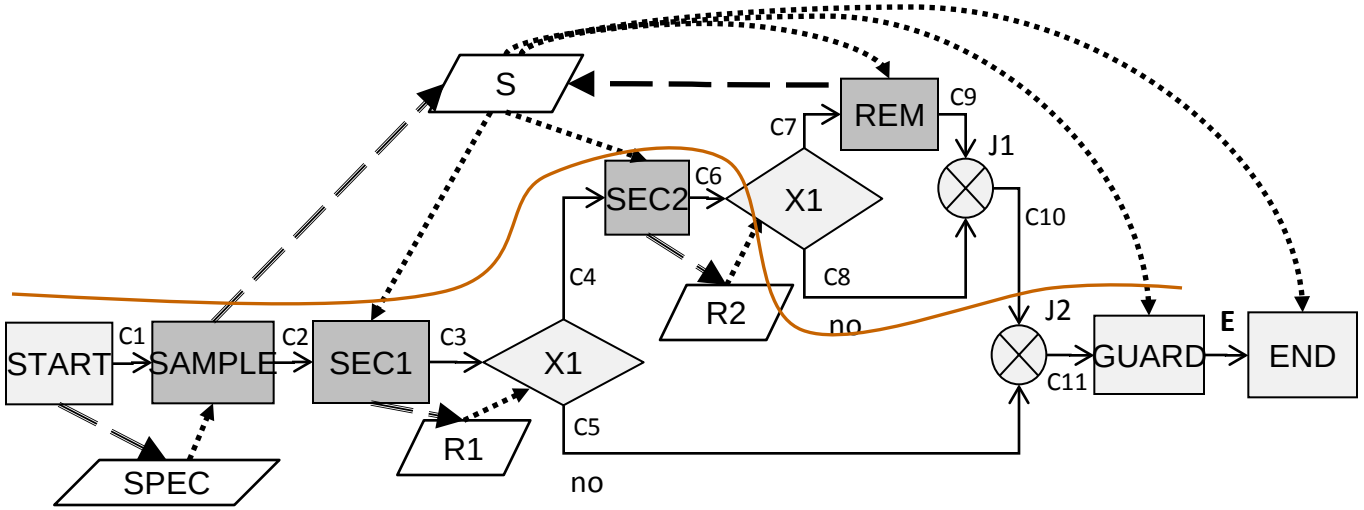


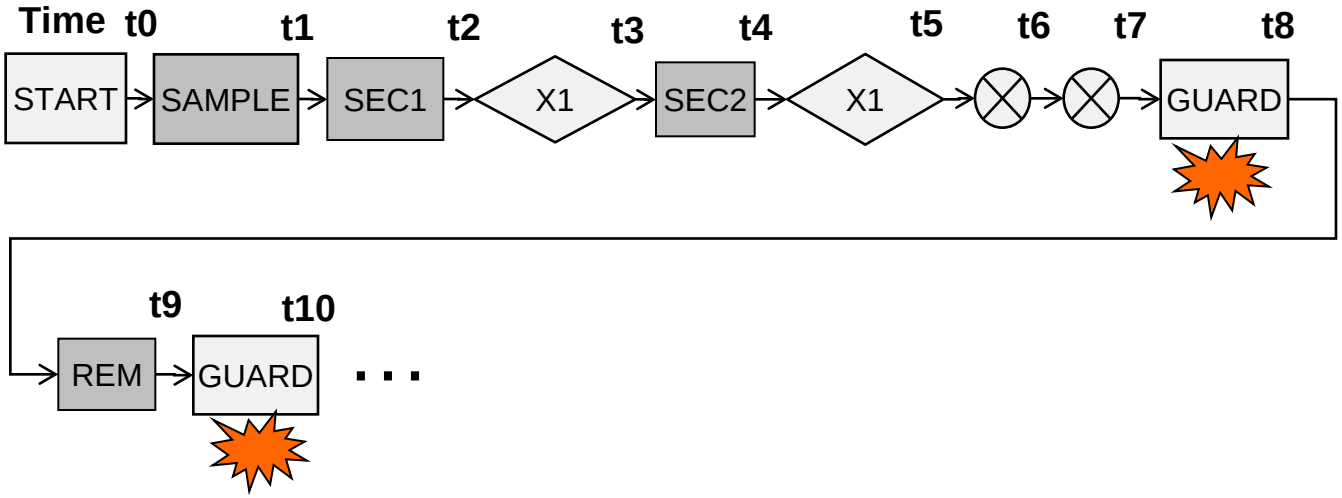
$[SAMPLE_{t1}]$

$[SEC2_{t4}]$

~~$[SEC1_{t2}]$~~

Diagnosis of process trajectories …





[$SAMPLE_{t1}$]

[$SEC2_{t4}$]

[$SAMPLE_{t1}$] [$SEC2_{t4}$ REM_{t9}]

Diagnosis problem (basic idea)

Given:

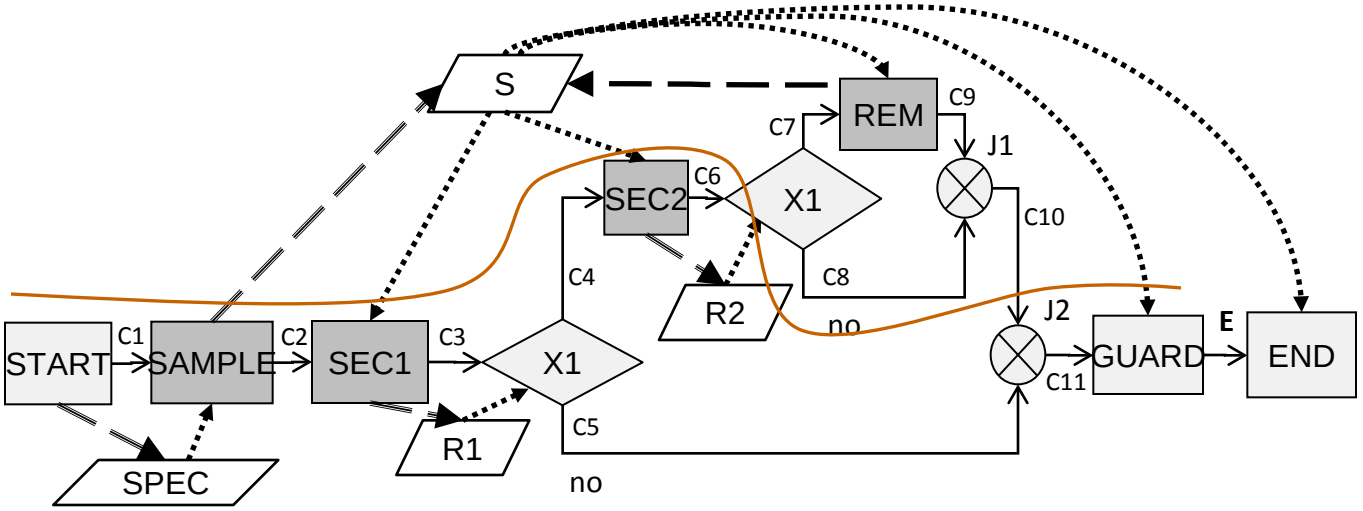
- Process definition (activities, variables)
- Observed input / output behavior of activity executions **OBS**
- Partially defined correct input / output behavior for activities
- Constraints on the correct behavior of activities, e.g.
  - For every input vector there must **exist** an output vector
  - For every input of an XOR-split either Output1 or Output2 is activated
  - Some activities behave deterministic (nondeterministic)
  - …

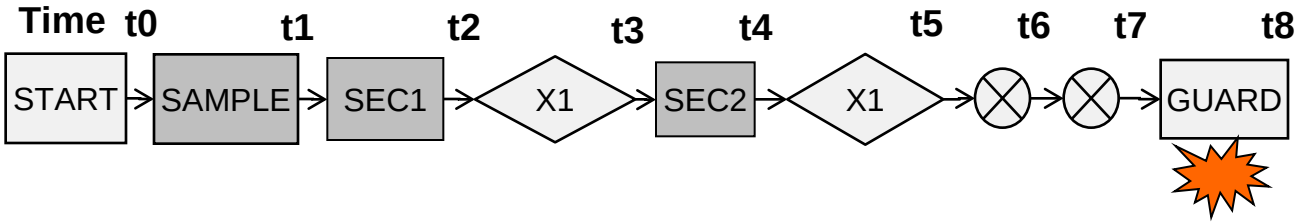
Diagnosis problem cont.

Diagnosis Δ is a subset of OBS (observed input / output behavior) iff there exists a correct behavior description of the activities s.t.

- OBS – Δ is part of the correct activity behavior
- The correct behavior of activities is consistent with the constraints on the correct behavior
- The correct behavior does not throw exceptions

Diagnosis of process trajectories …





$[SAMPLE_{t1}]$

$[SEC2_{t4}]$

~~$[SEC1_{t2}]$~~

Implementation

- Currently Eclipse 6.1
- Existential quantification replaced by symbolic values
- Unfolding of loops

Result: Improved precision of diagnosis cmp. to current methods

However

- ASP should provide much shorter, comprehensible descriptions of constraints on the correct behavior
- Proper handling of existential quantification should replace problem specific introduction of symbolic values

Grand challenge:

Integration of diagnosis and repair

This is not just:

1. Determine diagnoses
2. Compute a conditional plan s.t. a successful repair is guaranteed

Why?

- Every diagnosis corresponds to a plausible world

  (Note: For making predictions only the most probable worlds are considered)

- Every action could change the set of plausible worlds and therefore whether or not an object state is considered as correct

Consequences

Frame problem / ramification problem:

Given an action (e.g. execute a guard) and observing the outcome

What can we say about the correctness of object states?

- Compute the set of plausible (most probable) diagnoses
- Asses the plausibility of the correctness of object states
- For assessing the consequences of an action execution a diagnosis step is required
- Even computing the next minimal diagnosis for propositional Horn clause behavior descriptions is NP-hard
- Consequently, highly expressive action languages are required

With ASP and recent extensions you are on the right track.

Structure

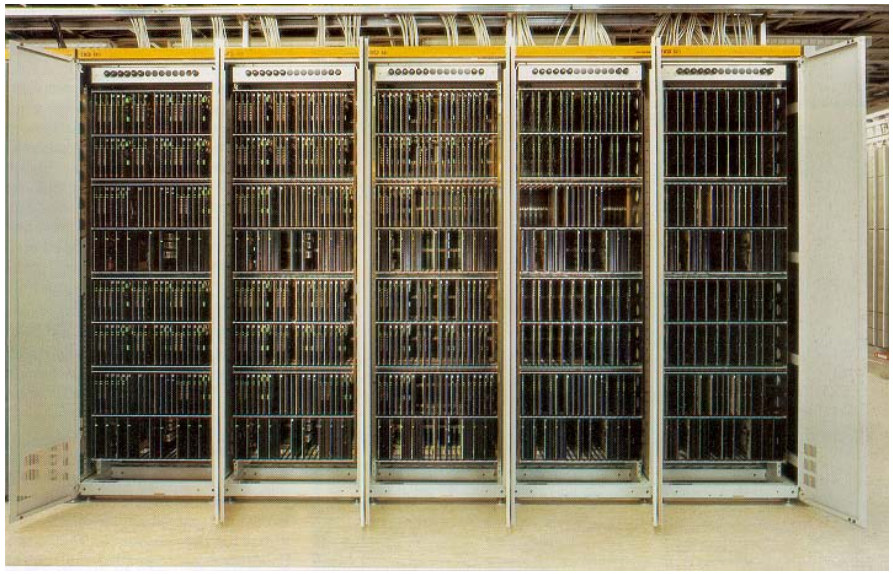
- Repair
- Diagnosis
- Configuration

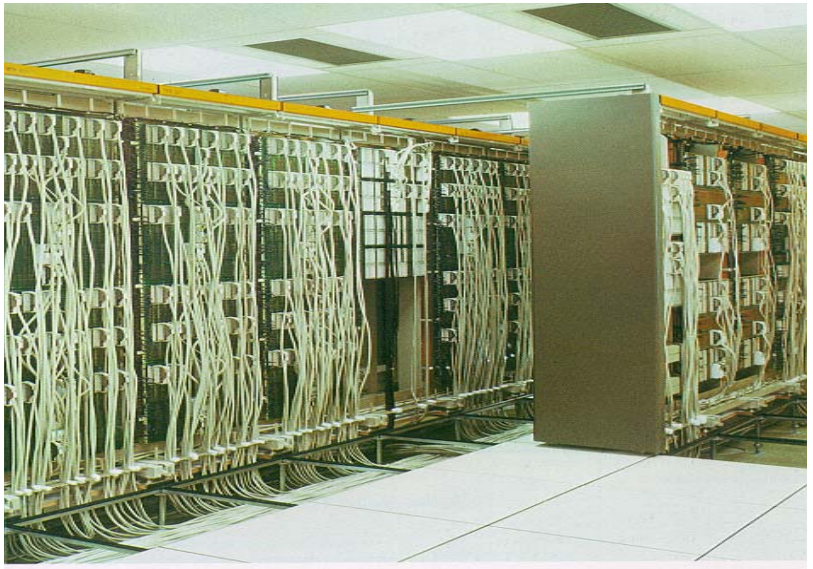
Configuration

Considered to be solved by many AI academics.

Solved ?

Telecommunication systems



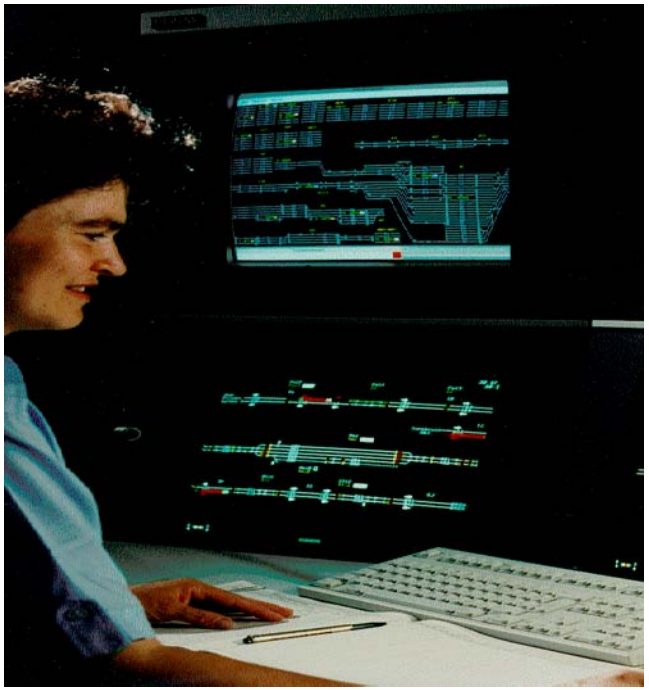


© Siemens

Large systems contain:

- 200 racks
- 1.000 frames
- 30.000 module
- 10.000 cables
- 2.000 other parts

Railway safety systems



© Siemens

Configuration of
- Hardware
- Software
- User interface

High development costs

Configuration problem

Customer requirements:

different for each instance (e.g. set of facts)

Configuration constraints:

stable for some time span (e.g. logical sentences)

Solution:

subset of a logical model,
as cost efficient as possible, given a time limit for computation

Myths

- Configuration problems are always under-constraint

  All popular SAT solvers fail on a simple example provided by Siemens.

- Configuration problems are static,
  i.e. no existential quantification needed

  Can current grounding methods handle large configurations?

- If we cannot find a solution then we change the problem

  You need very good arguments for changing the design of a system.

Configuration problems are still hard test cases.

Summary of KRR issues

Existential quantification

- required for configuration, diagnosis, planning

Reasoning costs versus expressivity

- preprocessing

Approximate solutions and
heuristics for the model generation process

- depending on solutions found so far

Handling of probabilities

- probabilities of logical models and sentences

Summary

After 20 years of implementing real world applications in domains like

- diagnosis and repair
- configuration
- recommender systems

answer set programming is very attractive.

ASP will have great influence on various fields in computer science.

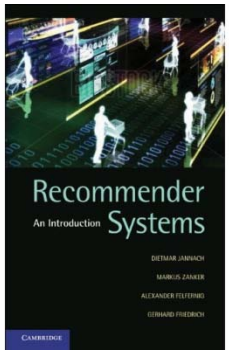
Literature

Repair: Friedrich G; Fugini M; Mussi E; Pernici B; Tagni G: Exception Handling for Repair in Service-Based Processes, IEEE Transactions on Software Engineering, 36 (2), 2010.

Diagnosis: Friedrich G; Mayer W; Stumptner M: Diagnosing Process Trajectories Under Partially Known Behavior, ECAI 2010.

Configuration: Fleischanderl G; Friedrich G; Haselböck A; Schreiner H; Stumptner M: Configuring Large Systems Using Generative Constraint Satisfaction. *IEEE Intelligent Systems,* Vol. 13, No. 4, July/August 1998, pp. 59-68.

Recommender systems, Jannach D; Zanker M; Felfernig A; Friedrich G; Cambridge University Press.



- Thank you -

Congratulations to the LP community, well done!

Happy anniversary