Applications of Artificial Intelligence

Sebastian Iwanowski
FH Wedel

Chapter 4:
Knowledge-Based Systems

4.3: Model-Based Reasoning
Details

Model-Based Diagnosis (MDS)

Terminology of the GDE approach:

<u>Component:</u>

Unit of which behaviour should be classified („diagnosed“)

usually enumerated from 1 to n

<u>Component type:</u>

collects components of same behaviour

<u>Behavioural mode:</u>

represents a specific behaviour of all components of that type

usually enumerated from 1 to k:

1 represents ok

2 thru k are the fault modes (ordered by probability)

<u>(Diagnosis) Candidate:</u>

Assignment of exactly one behavioural mode to each component of the system

Model-Based Diagnosis (MDS)

Terminology of the GDE approach:

Candidate:

(2 1 3 1 1 2 1) means: Component Nr. 1 is in behavioural mode 2
Component Nr. 2 is in behavioural mode 1
Component Nr. 3 is in behavioural mode 3
Component Nr. 4 is in behavioural mode 1
Component Nr. 5 is in behavioural mode 1
Component Nr. 6 is in behavioural mode 2
Component Nr. 7 is in behavioural mode 1

Conflict:

Assignment of exactly one behavioural mode to some components of the system

(0 1 0 0 0 2 0) means: Component Nr. 2 is in behavioural mode 1
Component Nr. 6 is in behavioural mode 2
About the other components no proposition is made.

Interpretation: It is not consistent that component 2 is in behavioural mode 1 and und component 6 is in behavioural mode 2.

Model-Based Diagnosis (MDS)

Terminology of the GDE approach:

Diagnosis (= consistent candidate):

Candidate not containing any conflict

Examples: (2 1 3 1 1 2 1) contains the conflict (0 1 0 0 0 2 0), i.e., it is not a diagnosis.

If (0 1 0 0 0 2 0) is the only conflict, then (1 1 1 1 1 1 1) is a diagnosis.

If (0 1 0 0 0 2 0) and (1 1 0 0 0 0 0) are the only conflicts, then (1 2 1 1 1 1 1) is a diagnosis

Preference between candidates:

A candidate A is preferred to another candidate B, if A assigns at most the number of the behavioural mode of B for each component.

Example: (1 1 1 1 1 1 1) is preferred to (1 2 1 1 1 1 1)

Maximum preferred diagnosis:

A diagnosis is called a maximum preferred diagnosis, if all preferred candidates contain conflicts, i.e. the diagnosis is maximum with respect to the preference relation.

Example: If (0 1 0 0 0 2 0) and (1 1 0 0 0 0 0) are the conflicts, then (1 2 1 1 1 1 1) and (2 1 1 1 1 1 1) are the only two maximum preferred diagnoses.

Model-Based Diagnosis (MDS)

Goal of MDS (Daimler enhancement of the GDE):

1) **Base functionality:** **Find the best diagnoses**

2) **Extended functionality:** **Repair instruction:**
Propose actions and tests in order to distinguish between diagnoses found in 1)

Details of 1): Find the maximum preferred diagnoses.

If there are too many maximum preferred diagnoses, the focus should be restricted to the *most probable* ones among *all* maximum preferred diagnoses. The remaining maximum preferred diagnoses are to be marked as *pending* and may be inserted into focus at a later time.

Possible focus restriction policies (may be combined):

a) **Determine a maximum number of focus diagnoses**

b) **Determine a probability threshold for the gap between focus diagnoses and pending diagnoses.**

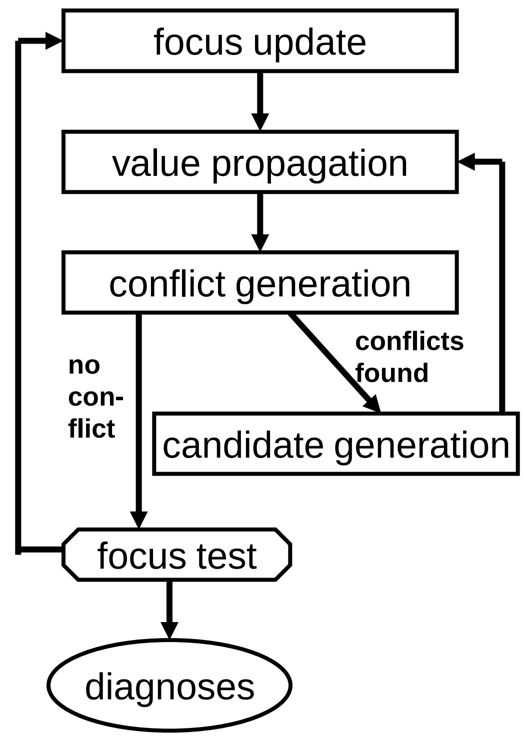
Model-Based Diagnosis (MDS)

Algorithm for finding the most probable maximum preferred diagnoses (Problem 1):

At any time, all candidates of the focus are maximum preferred.

1. **Update the focus candidates: Initialise with 11….1.**
   At later stages, pending candidates may be dragged into focus.
2. **Generate and propagate all values resulting from behavioural modes of candidates in focus.**
3. **Find the minimal conflicts from the propagated values.**
4. **Exclude the candidates containing conflicts and compute new maximum preferred candidates not containing any conflict.**
5. **If focus is sufficiently large, the goal is achieved.**
   Otherwise continue with 1.

Diagnostic cycle



In reality, steps 2 thru 4 are implemented concurrently.

(achieved by event oriented programming)

In the following, the methods for **candidate generation** and **conflict generation** are described separately.

MDS: Candidate generation

INPUT:

- **Old conflicts and all maximum preferred und consistent diagnoses for these conflicts**
- **New conflicts**

OUTPUT:

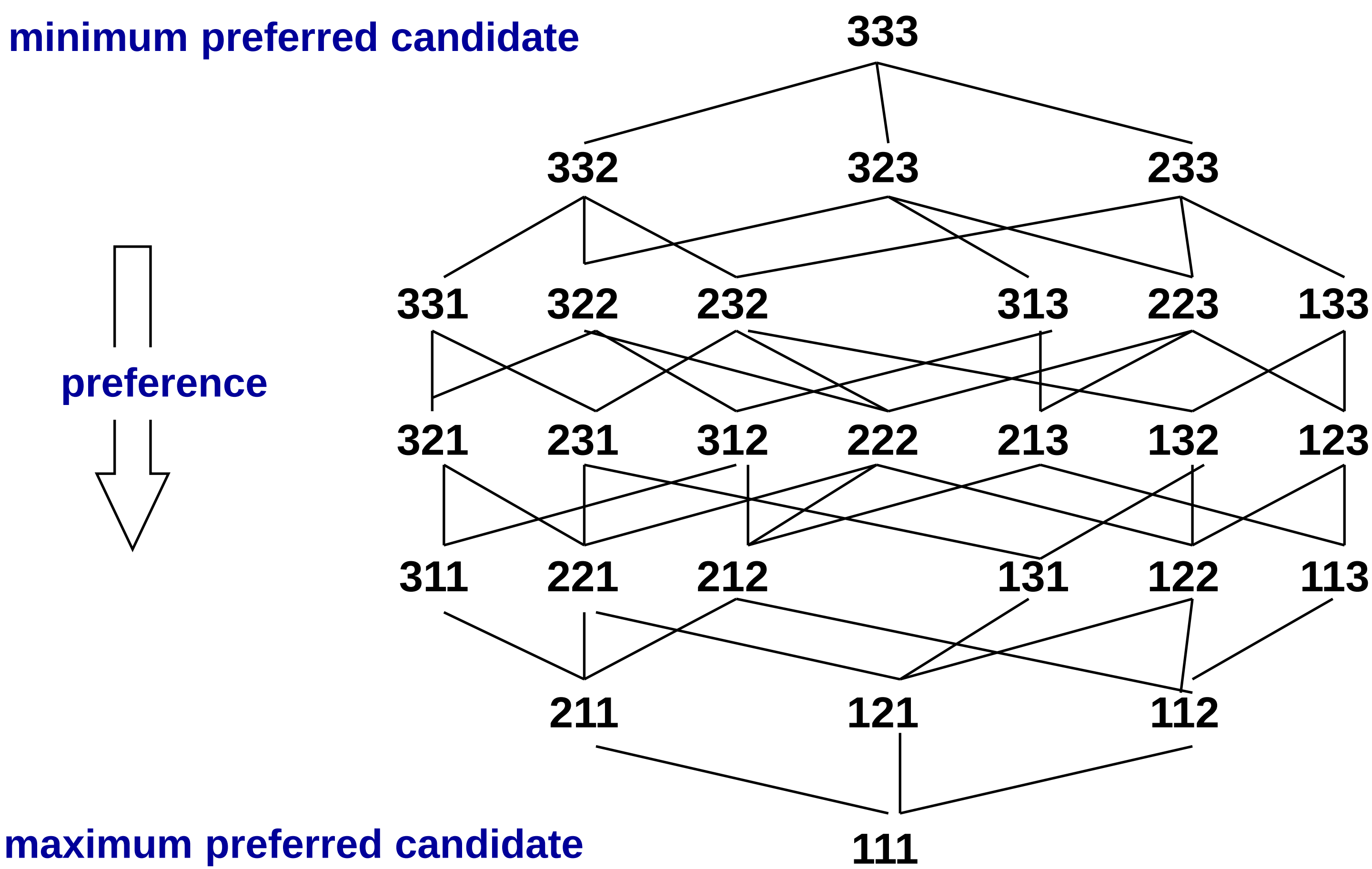
- **Set of maximum preferred candidates being consistent for the new conflicts, too**

Embedding the candidate generation into the diagnostic process:

- Output of candidate generation will be taken as input in the next diagnostic cycle.
- Value propagation may find new conflicts.
- New conflicts may kick out diagnoses from focus.
- If no new conflicts are found, the diagnostic process is finished.

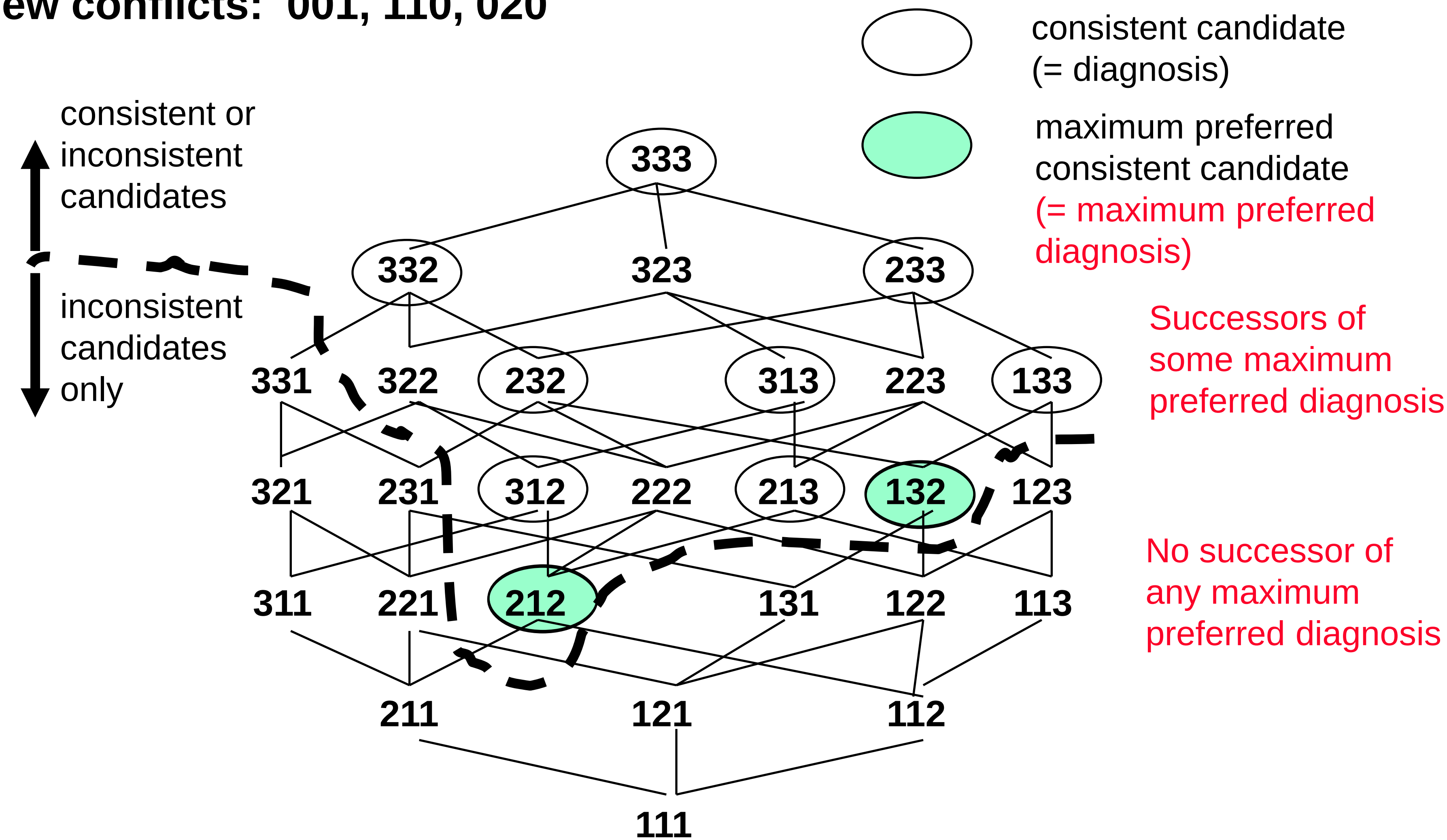
MDS: Preference web of candidates

Example: **3 components**
3 behavioural modes for each of the components



MDS: Preference web of candidates

New conflicts: 001, 110, 020



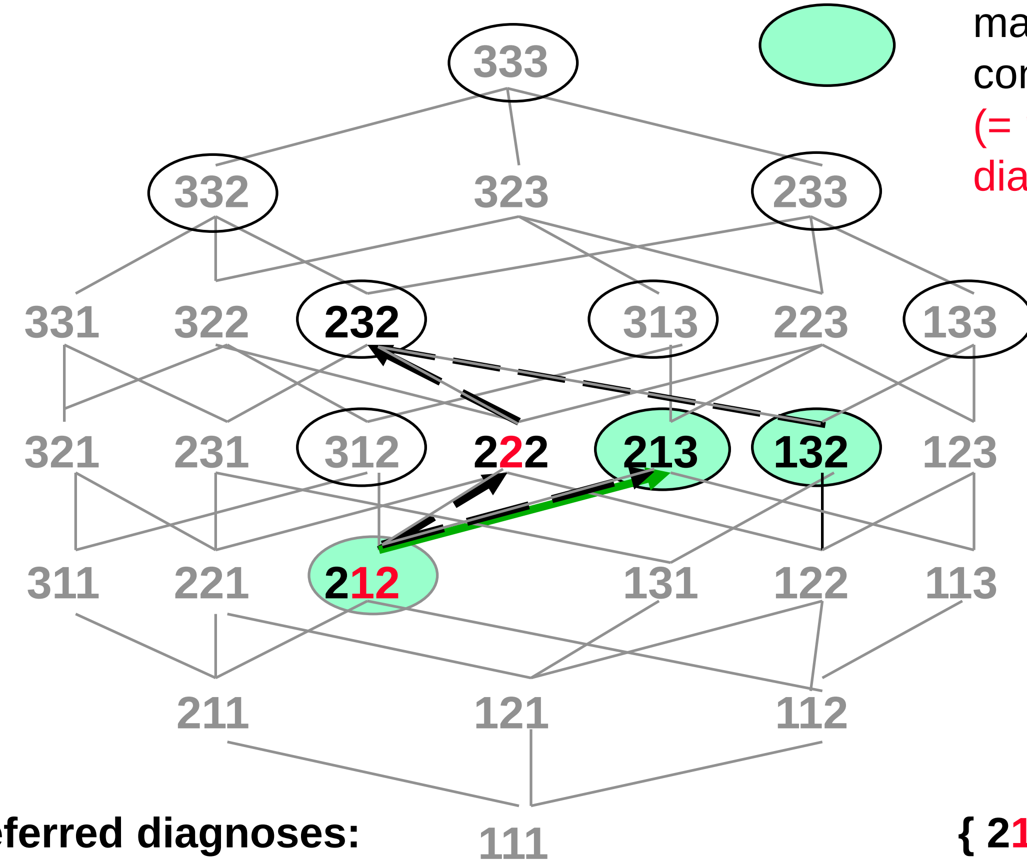
MDS: Candidates update

Old conflicts: 001, 110, 020

New conflict: 012

consistent candidate
(= diagnosis)

maximum preferred
consistent candidate
(= maximum preferred
diagnosis)



Former maximum preferred diagnoses:	**{ 212, 132 }**
New maximum preferred diagnoses (stage 1):	**{ 222, 213, 132 }**
New maximum preferred diagnoses (stage 2):	**{ 213, 132 }**

MDS: Candidates update

Actions at detection of a new conflict:

1) Consistency check of all maximum preferred diagnoses
2) Removal of all candidates proven to be inconsistent
3) Generation of the preference successors of each candidate just removed
4) Adopting the preference successors satisfying the following conditions:
    - The successor is not preferred by a different consistent diagnosis.
    - The successor is consistent itself.

MDS: Candidates update

Actions at detection of a new conflict :

3) Generation of the preference successors of each candidate just removed:

- If C is a conflict contained in an old diagnosis, then generate only successors of C changing the behavioral mode of **just one component contained in C**.

  (Generation of direct successors only, directly referring to conflict C)

> ***Remark: This restricted method does not skip any eventual diagnosis***
>
> ***Prop.:*** Each successor diagnosis not containing C is successor diagnosis of a direct successor not containing C

- If one of the direct successors contains a conflict C', then do not generate this successor, but rather all successors referring directly to C'.

MDS: Optimising the candidate generation

Eliminating irrelevant conflicts:

- Conflicts are only relevant, if they may eventually remove a successor of a presently maximum preferred diagnosis.
- For the consistency test, only consider relevant conflicts: Each diagnosis d stores the relevant conflicts. Any successor of d will only be checked for the conflicts of d’s list.

Examples for relevant conflicts:

conflict:	**0 2 2**	**2 0 2**	**2 0 2**
candidate:	**3 1 2**	**2 1 1**	**1 1 3**
relevant ?			

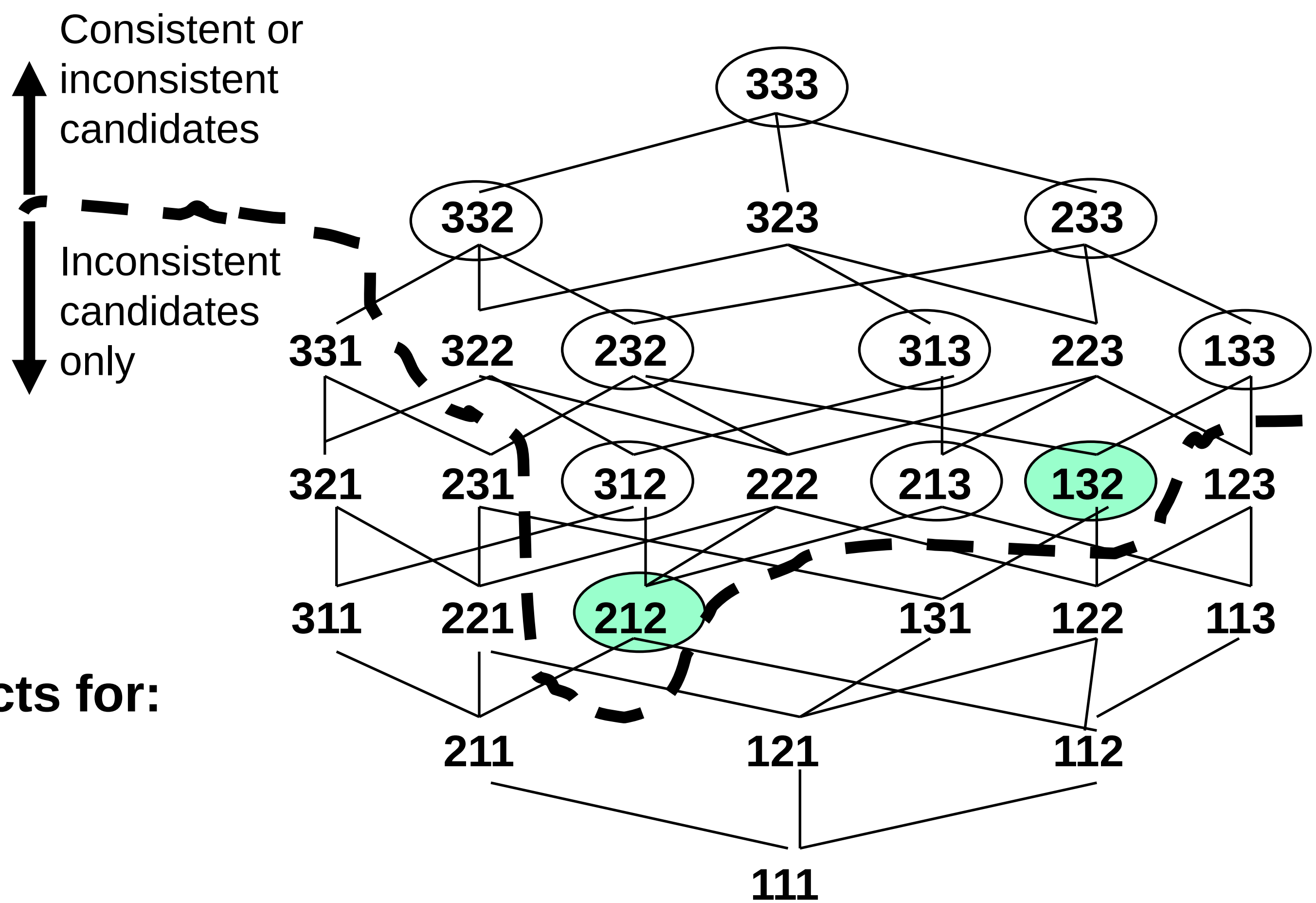
Mathematical criterion for the relevance of a conflict

(easy to check!)

A conflict c is relevant for a diagnosis d if for all components holds: c either assigns no mode (0) or a mode at least as high as the mode in d.

MDS: Optimising the candidate generation

Eliminating irrelevant conflicts:



conflicts:
{001, 110, 020}

relevant conflicts for:
- **212: 020**
- **132: none**

This means that 001 and 110 will never be checked again!

MDS: Optimising the candidate generation

The Daimler product MDS contains a lot of further optimisations for accelerating the candidate generation process which are not mentioned here.

MDS: Conflict generation

Candidate generation solves the following task:

- **Given a set of conflicts: Find the most probable maximum preferred diagnoses taking into account those conflicts.**
- **This reduces the problem of finding the best diagnosis to the following task: Find the set of conflicts !**

What is a conflict ?

- **Assignment of exaclty one behavioural mode to some component of a system**
- **Logically, a conflict is a disjunction of negative literals.**
- Comparing: Logically, a diagnosis is a conjunction of positive literals.

How is a conflict generated?

- **by values contradicting each other**
- **The contradicting values are backed by different assumptions.**
- **Then one of the assumptions must be false.**

TMS: Truth Maintenance System

Objects of a TMS:

Propositional node:

Represents an arbitrary proposition (may be true or false)

Justification:

$A_1 \land A_2 \land \ldots \land A_n \Rightarrow C$ where $A_1, A_2, \ldots, A_n$, C are propositional nodes

$A_1, A_2, \ldots, A_n$ are the antecedents of the justification

C is the conclusion of the justification

Contradiction node ($\perp$):

represents a proposition which holds by no means

TMS: Truth Maintenance System

Propositional nodes

Justifications

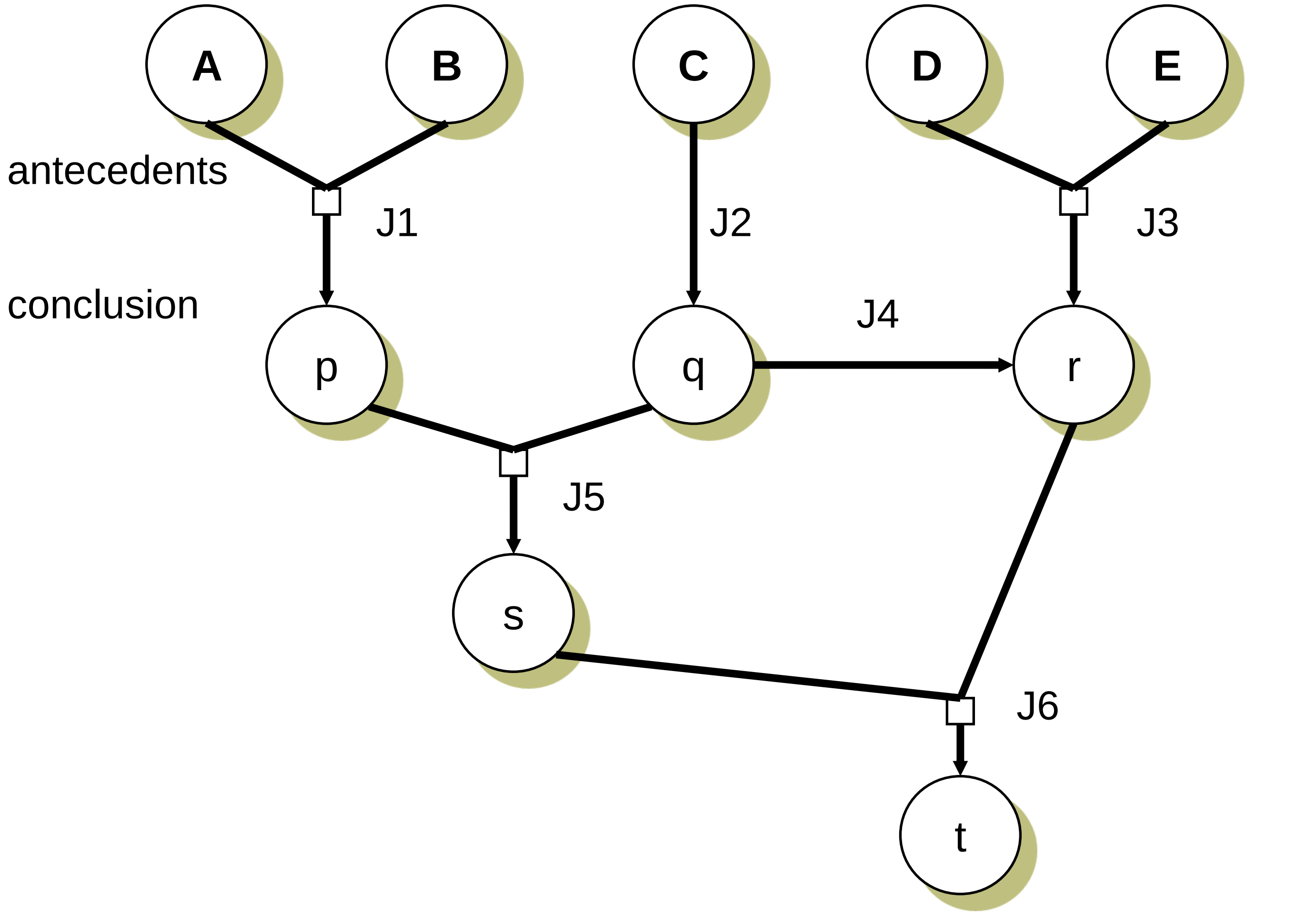
Propositional nodes

Justification

Propositional node

Justification

Propositional node



From the combination of propositions,
a justification makes a new proposition.
The antecedents of a justification are to be considered as conjunction.

ATMS: Assumption-based Truth Maintenance System

Functionality of a general TMS:

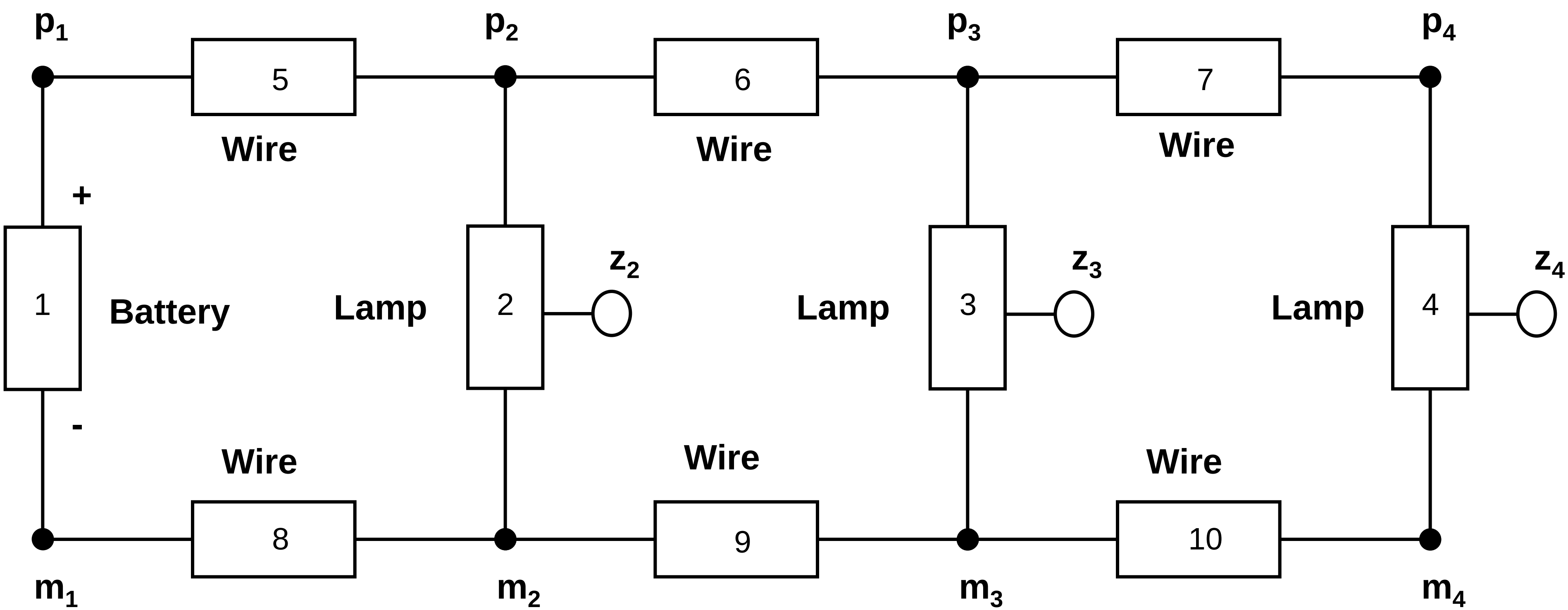
1) Certain propositional nodes are considered true (beliefs).
2) TMS determines by propagation of these assumptions via the justifications which other propositions must also hold then.
3) In particular, if the contradiction node must hold, then the assumptions must be contradictory.

Additional functionality of an ATMS:

- An ATMS works with *several* assumption sets in parallel: A (context) environment is the set of assumptions that should hold at the same time, but there may be different such environments holding alternatively.

1) The propositions are assigned with the assumption environments under which they must hold.
2) The ATMS propagates these assumtion environments over the justifications and determines which other propositions must hold then as well.
3) In particular, the environments of the contradiction node reveals which enviroments are contradictory.

Example for applying an ATMS



Behavioural modes:

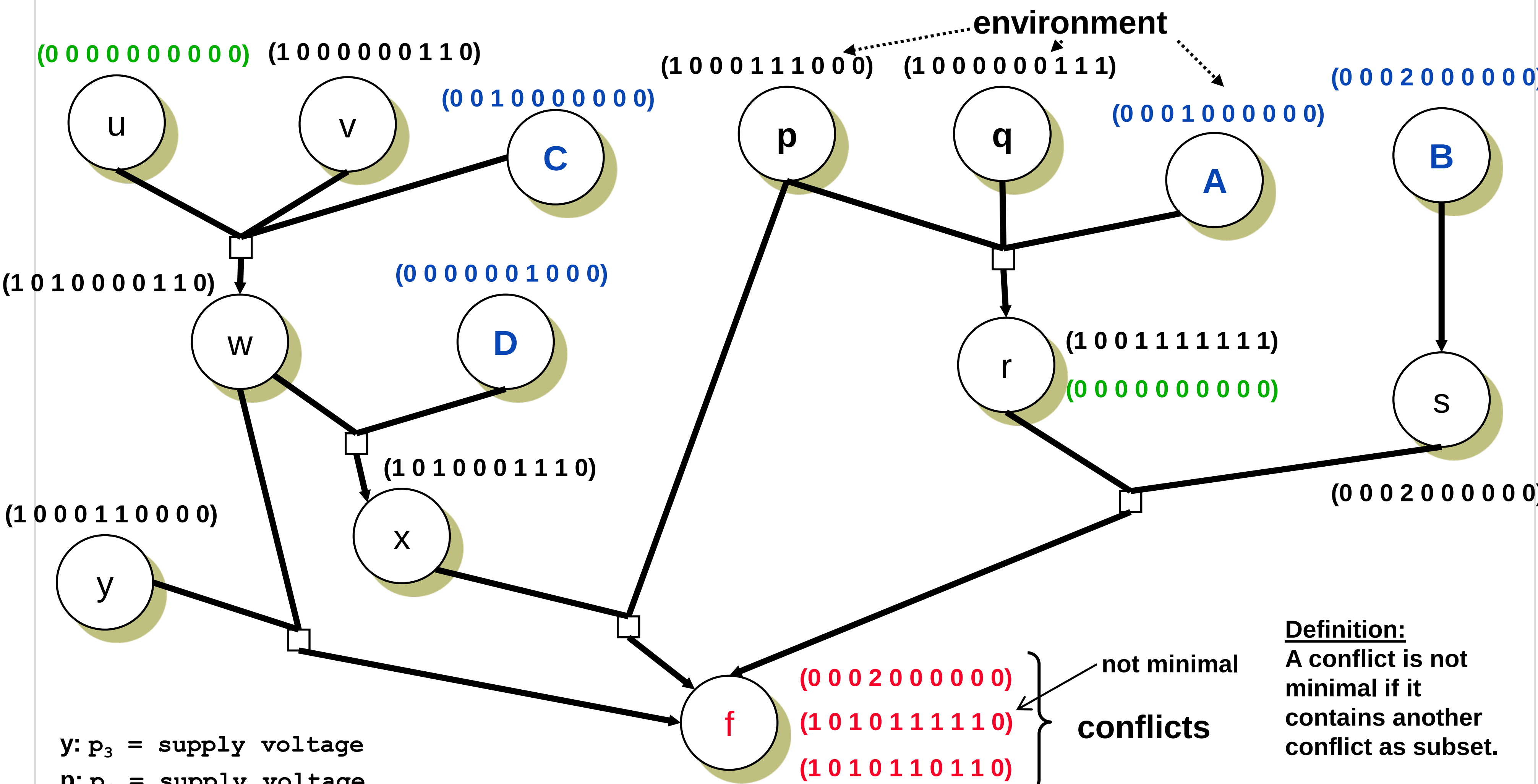
Mode 1 for all component types: normal behaviour

Modus 2 for all component types: unique fault mode

Battery: Modus 2 ⇒ (minus = `ground voltage`)

Lamp: Modus 2 ⇒ (z = `dark`)

Example for applying an ATMS



y: p_3 = supply voltage

p: p_4 = supply voltage

q: m_4 = ground voltage

A: Component 4 is ok

r: z_4 = lit

B: Component 4 is faulty

s: z_4 = dark

f: ⊥ (contradiction)

u: z_3 = dark

v: m_3 = ground voltage

C: Component 3 is ok

w: p_3 = ground voltage

D: Component 7 is ok

x: p_4 = ground voltage

ATMS: Assumption-based Truth Maintenance System

Terminology of ATMS:

Propositional node:

The propositional nodes distinguish between *normal propositions* and *assumptions*, i.e. the class of assumption nodes is a specialisation of propositional nodes.

Environment:

Context of assumptions: *Conjunction* of assumptions, under which a proposition holds (if all assumptions of this environment are valid)

Label:

Set of different environments for a propositional node. Different environments need not be consistent to each other. The proposition holds already under the *disjunction* of the environments.

conflict (nogood):

Environment of the label of the contradictory node

ATMS: Assumption-based Truth Maintenance System

Application of an ATMS for model-based diagnosis:

Propositional nodes:

1) **„Normal“ nodes: Assignment of a certain value to a certain position (variable) in the system**

2) **Assumption node: Assignment of a behavioural mode to a component**

Justification: **Application of a generic behavioural rule to actual values**

Environment:

Concurrent (conjunction) assignment of behavioural modes to components under which a proposition would hold. The assignment need not be complete, i.e. it is an arbitrary candidate (like in a conflict).

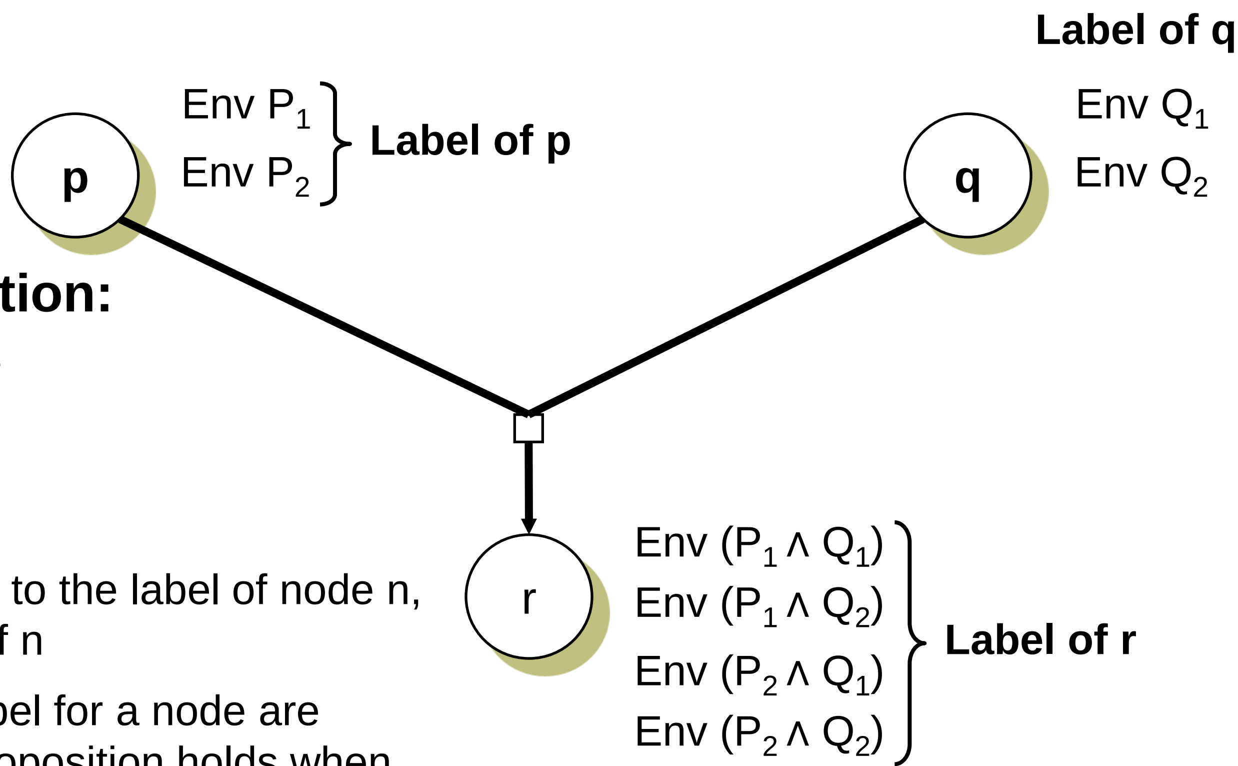
For assumption nodes: Assignment of a behavioural mode to exactly one component

conflict (nogood):

Environment of the label of the contradictory node: Assignment of behavioural modes to components of which at least one must be faulty.

This enables the same notation and meaning of conflicts as in the terminology of the GDE.

Label update in an ATMS



Interpretation of justification:

- r holds when q and r are true (conjunction)

Interpretation of labels:

- When environment e belongs to the label of node n, this means: e ⇒ proposition of n
- Several environments of a label for a node are treated as disjunction: The proposition holds when at least one of the environments is true.

Elimination of redundant environments:

- Contradictory environments may be removed.
- This enables the removal of all environments containing conflicts.
- Environments implying other environments of the same label may be omitted as well.

User interface of an ATMS

Input of problem solver:

- Assumption nodes
- “Normal” nodes
- Justifications between the nodes
  (they must be obtained from the component library applied to actual values)
- Certain environment assignments to normal nodes,
  e.g., observations or other premises as (0 0 ... 0)

Output to the problem solver:

- Set of minimal conflicts (Definition of minimality on slide 21)

The ATMS performs automatically:

These are a lot of operations !

- Generation of labels for the assumption nodes
- Update of labels for all conclusions where the label of
  some antecedent has changes.
- Elimination of redundant environments

User interface of an ATMS

Input of problem solver:

- Assumption nodes
- “Normal” nodes
- Justifications between the nodes
  (they must be obtained from the component library applied to actual values)
- Certain environment assignments to normal nodes,
  e.g., observations or other premises as (0 0 ... 0)

Output to the problem solver:

- Set of minimal conflicts

Interaction with the candidate generator:

- Generate all assumption nodes for the focus diagnoses
- Value propagation (simulation):
  Compute all values resulting from assumptions of the focus diagnoses,
  generate the respective propositional nodes und justifications,
  plug this into the ATMS.
- Ask the ATMS for the new conflicts.

Value propagation and ATMS

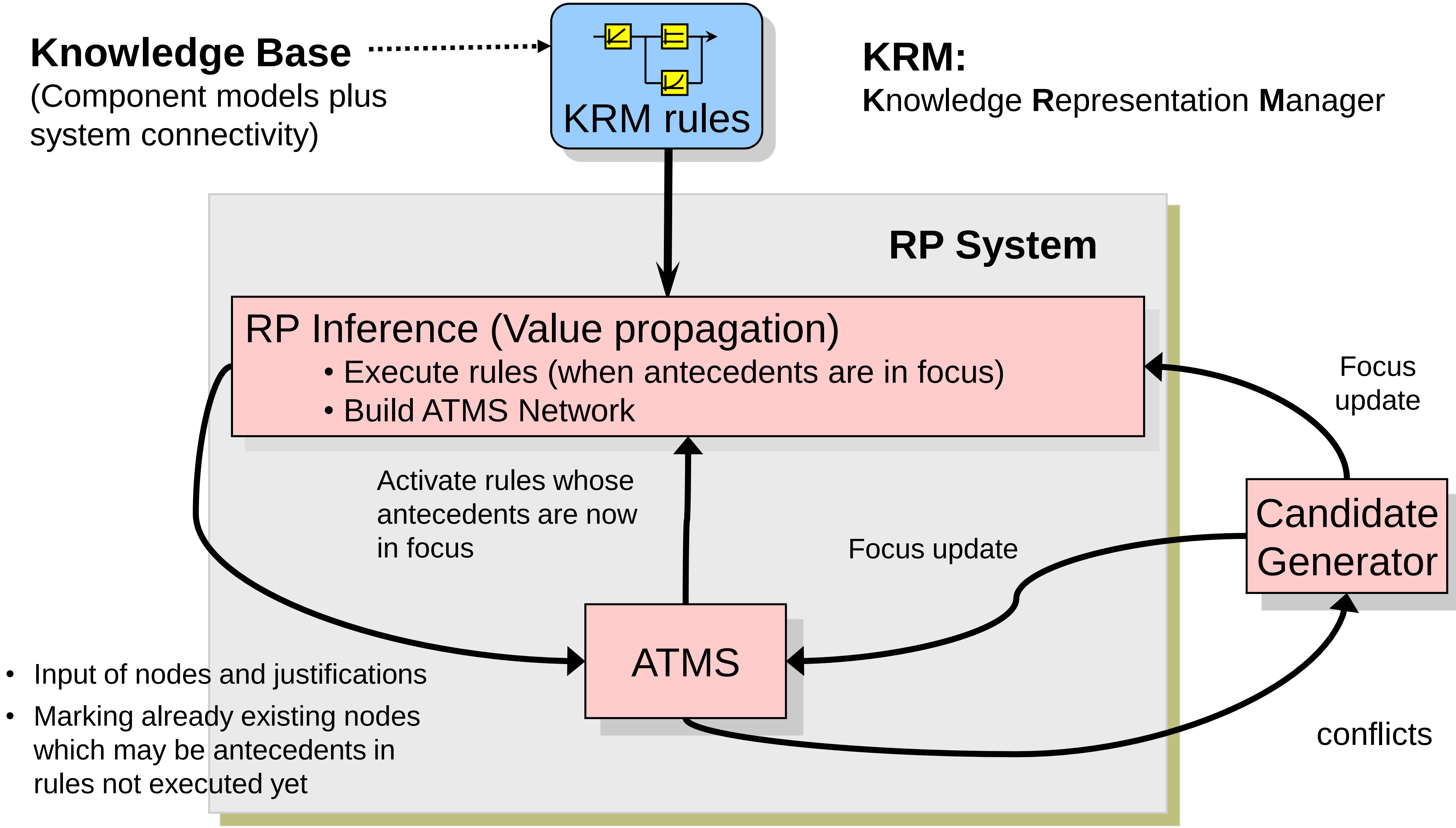
What is propagation in general ?

- Propagation is the distribution of information in a network made of nodes and edges

Separation of value propagation (RP) and ATMS:

- The **ATMS** is responsible for *propagation of environments* in a given network with already determined value dependencies.
- The *propagation of values* is performed by a rule propagator (**RP**) which generates justifications for actual values from the generic values of the behavioural modes of the components. Thus, RP generates the network of value dependencies required by the ATMS.

Value propagation and ATMS



In optimised candidate generators and ATMS's the interface is more complicated.

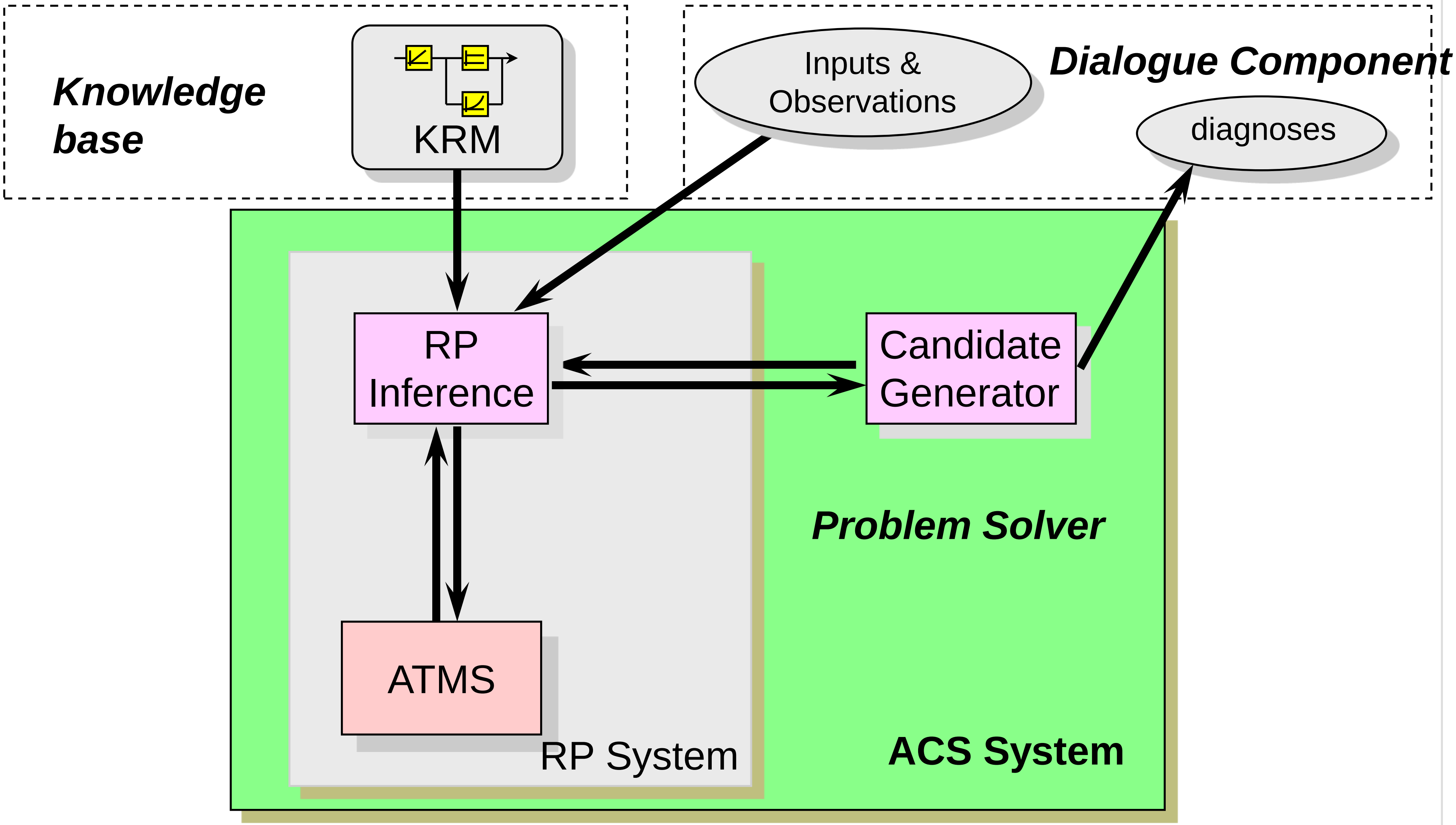
Value propagation and ATMS

What is the benefit of separating value propagation and ATMS?

→ **Better software architecture by modularisation**

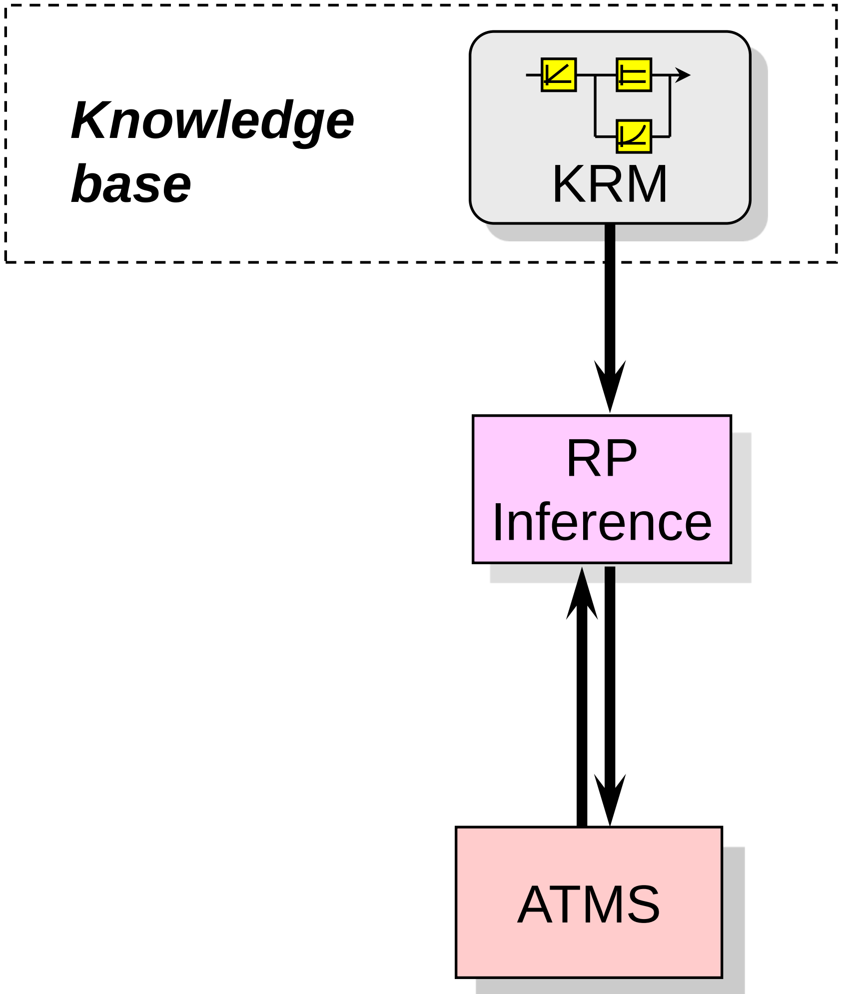
- **Values** are generated mostly from observations (measurements) and intended actions. This is not frequent, thus, there are **not many** values to be considered.
- **Environments** are generated from assumptions about behavioural modes. Of such constructs there exist **a lot of** (even at single faults at least as many as there exist components).
- This makes the update of focus environments much more often to occur than the computation of new values. The update of focus environments may be considered an ATMS internal problem

Interaction of candidate generator, RP and ATMS



ACS: **A**ssumption-based **C**onstraint **S**olver

Requirement to the knowledge base



What does the knowledge base have to provide to the inference component (problem solver)?

- Rules for the relations of values in each behavioural mode (*component models*)
- Knowledge about the value domains:
  When are two values considered contradictory?

Daimler's MDS solves these requirements by offering a constraint language for component models.