Künstliche Intelligenz

Sebastian Iwanowski
FH Wedel

Kap. 4:
KI-Architektur: Wissensbasierte Systeme

4.4: MDS: Funktionsweise der Inferenzmaschine (Basisfunktionalität)

Modellbasierte Diagnose

Begriffswelt GDE:

Komponente:

Einheit, deren Verhalten diagnostiziert werden soll

üblicherweise nummeriert von 1 bis n

Komponententyp:

fasst Komponenten gleichartigen Verhaltens zusammen

Verhaltensmodus:

Einem Komponententyp zugeordnete Verhaltensbeschreibung

üblicherweise nummeriert von 1 bis k:

1 steht für ok

2 bis k sind die Fehlermodi (geordnet nach Wahrscheinlichkeit)

(Diagnose-)Kandidat:

Zuweisung von genau einem Verhaltensmodus an jede Komponente des Systems

Modellbasierte Diagnose

Begriffswelt GDE:

Kandidat:

(2 1 3 1 1 2 1) bedeutet: **Komponente Nr. 1 ist in Verhaltensmodus 2**
Komponente Nr. 2 ist in Verhaltensmodus 1
Komponente Nr. 3 ist in Verhaltensmodus 3
Komponente Nr. 4 ist in Verhaltensmodus 1
Komponente Nr. 5 ist in Verhaltensmodus 1
Komponente Nr. 6 ist in Verhaltensmodus 2
Komponente Nr. 7 ist in Verhaltensmodus 1

Konflikt:

Zuweisung von genau einem Verhaltensmodus an einige Komponenten des Systems

(0 1 0 0 0 2 0) bedeutet: **Komponente Nr. 2 ist in Verhaltensmodus 1**
Komponente Nr. 6 ist in Verhaltensmodus 2
über die anderen Komponenten wird keine Aussage gemacht

Interpretation: **Es ist unvereinbar, dass sich Komponente 2 in Verhaltensmodus 1 und Komponente Nr. 6 in Verhaltensmodus 2 befindet.**

Modellbasierte Diagnose

Begriffswelt GDE:

Diagnose (= konsistenter Kandidat):

Kandidat, der keinen Konflikt enthält

Beispiele: **(2 1 3 1 1 2 1) enthält den Konflikt (0 1 0 0 0 2 0), ist also keine Diagnose**

Wenn (0 1 0 0 0 2 0) der einzige Konflikt ist, ist (1 1 1 1 1 1 1) eine Diagnose

Wenn (0 1 0 0 0 2 0) und (1 1 0 0 0 0 0) die Konflikte sind, ist (1 2 1 1 1 1 1) eine Diagnose

Präferenz zwischen Kandidaten:

Ein Kandidat A ist einem anderen Kandidaten B präferiert, wenn A für jede Komponente maximal den Verhaltensmodus von B zuweist.

Beispiel: **(1 1 1 1 1 1 1) ist präferiert zu (1 2 1 1 1 1 1)**

Präferierte Diagnose:

Eine Diagnose ist präferiert, wenn alle ihr präferierten Kandidaten Konflikte enthalten, sie also bezüglich der Präferenz maximal ist.

Beispiel: **Wenn (0 1 0 0 0 2 0) und (1 1 0 0 0 0 0) die Konflikte sind, sind (1 2 1 1 1 1 1) und (2 1 1 1 1 1 1) die beiden einzigen präferierten Diagnosen.**

Modellbasierte Diagnose

Ziel von MDS (Daimler-Weiterentwicklung der GDE):

Basisfunktionalität Diagnosefindung:

1) **Finde die wahrscheinlichsten präferierten Diagnosen !**
   (aus Komplexitätsgründen wird die Stückzahl stark begrenzt)

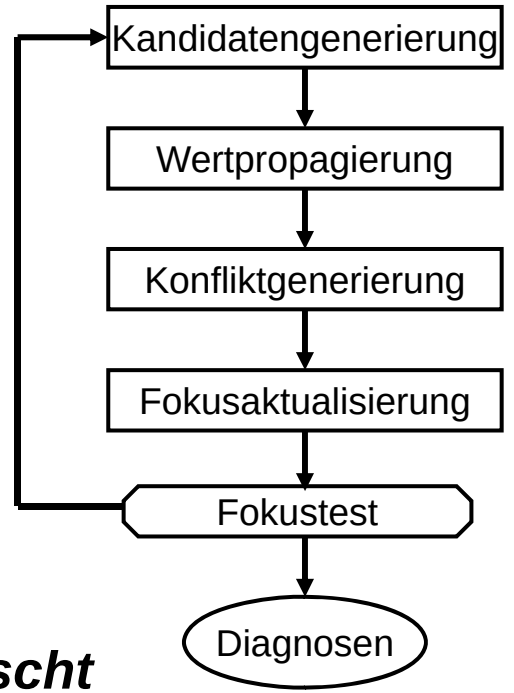
Erweiterte Funktionalität Reparaturanleitung:

2) **Schlage Aktionen und Tests vor, um die möglichen Diagnosen weiter einzuschränken !**

Modellbasierte Diagnose

Algorithmus zum Finden der wahrscheinlichsten präferierten Diagnosen (Aufgabenstellung 1):

1. **Nimm Kandidaten in den Fokus auf.**
2. **Generiere und propagiere alle Werte, die sich aus den Verhaltensmodi der Kandidaten im Fokus ergeben.**
3. **Finde die minimalen Konflikte aus den propagierten Werten.**
4. **Schließe die Kandidaten aus, die Konflikte enthalten.**
5. **Falls Fokus noch genügend groß, dann Ziel erreicht, anderenfalls weiter bei 1.**



In der Realisierung werden die Schritte 1 bis 4 vermischt

(erreicht durch ereignisorientierte Programmierung)

Im Folgenden werden die Verfahren für die **Kandidatengenerierung** und **Konfliktgenerierung** getrennt beschrieben.

MDS: Kandidatengenerierung

INPUT:

- **Alte Konflikte und die für diese Konflikte präferierten und konsistenten Kandidaten**
- **Neue Konflikte**

OUTPUT:

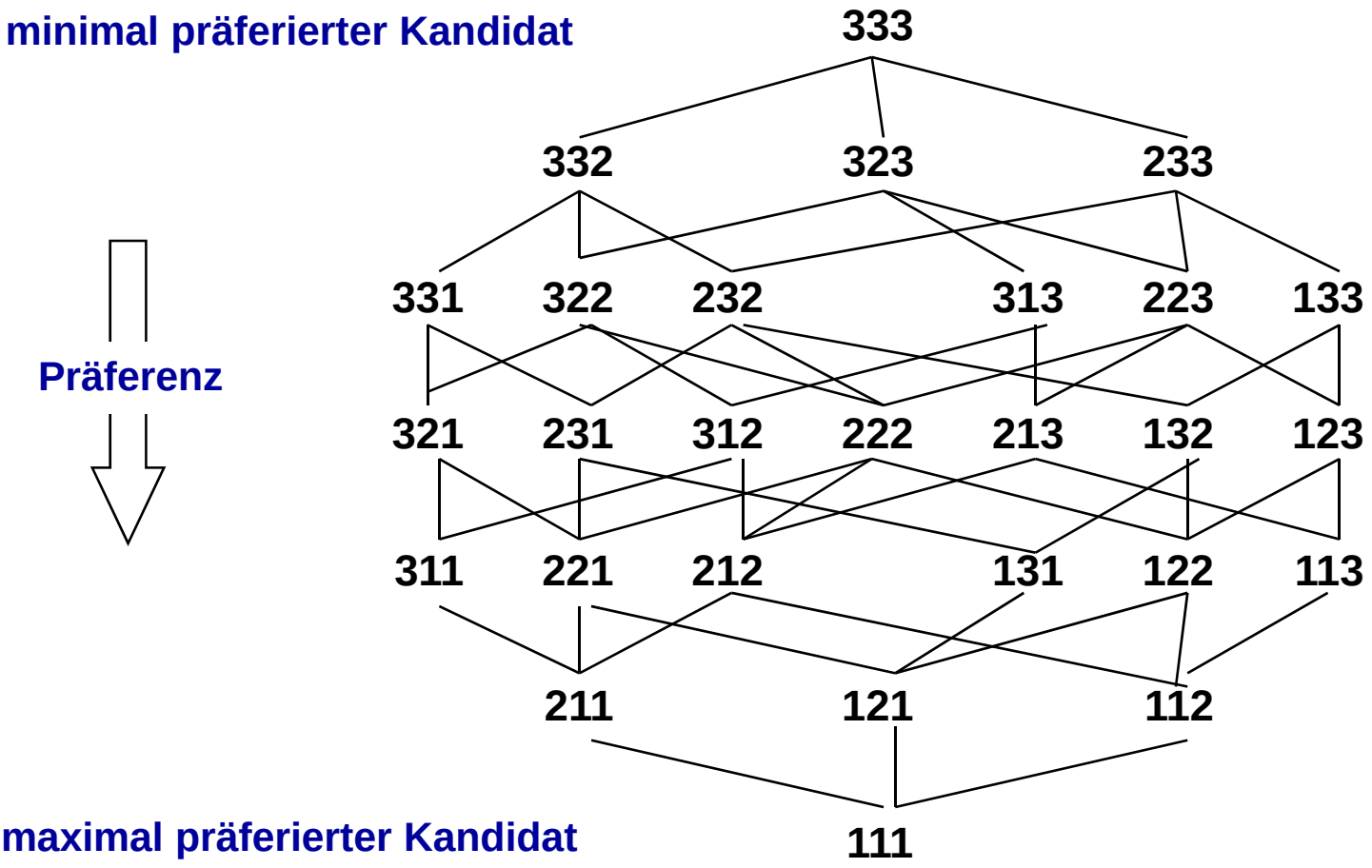
- **Menge der präferierten Kandidaten, die auch für die neuen Konflikte konsistent sind**

Einbettung der Kandidatengenerierung in den Diagnoseprozess:

- Output der Kandidatengenerierung wird in den Fokus genommen
- Durch Wertepropagierung werden neue Konflikte gefunden
- Diese Konflikte werden als Input für eine neue Runde der Kandidatengenerierung genommen
- Wenn keine neuen Konflikte gefunden werden, ist der Diagnoseprozess beendet.

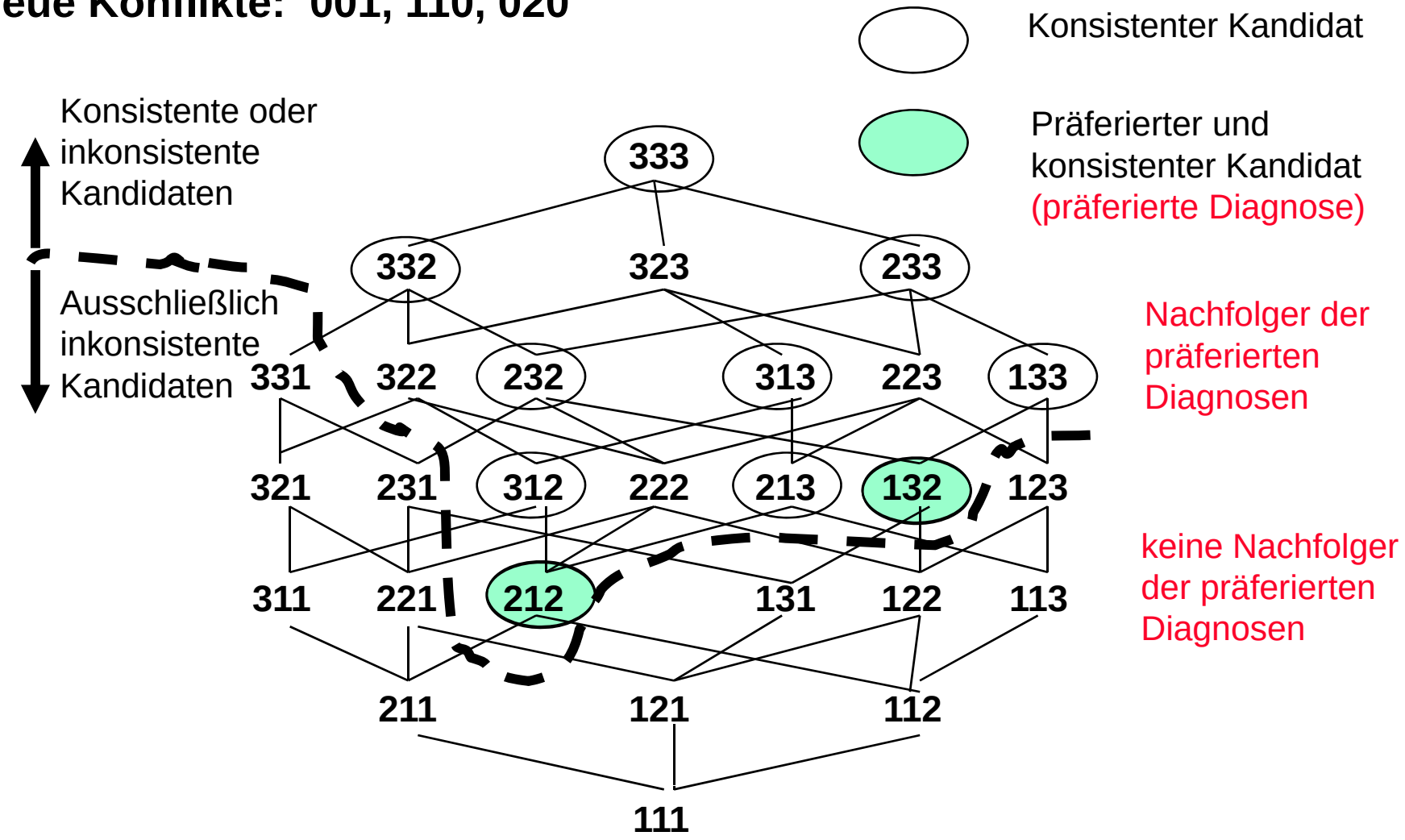
MDS: Die Kandidaten im Präferenznetz

Beispiel: **3 Komponenten**
Für jede Komponente 3 Verhaltensmodi



MDS: Die Kandidaten im Präferenznetz

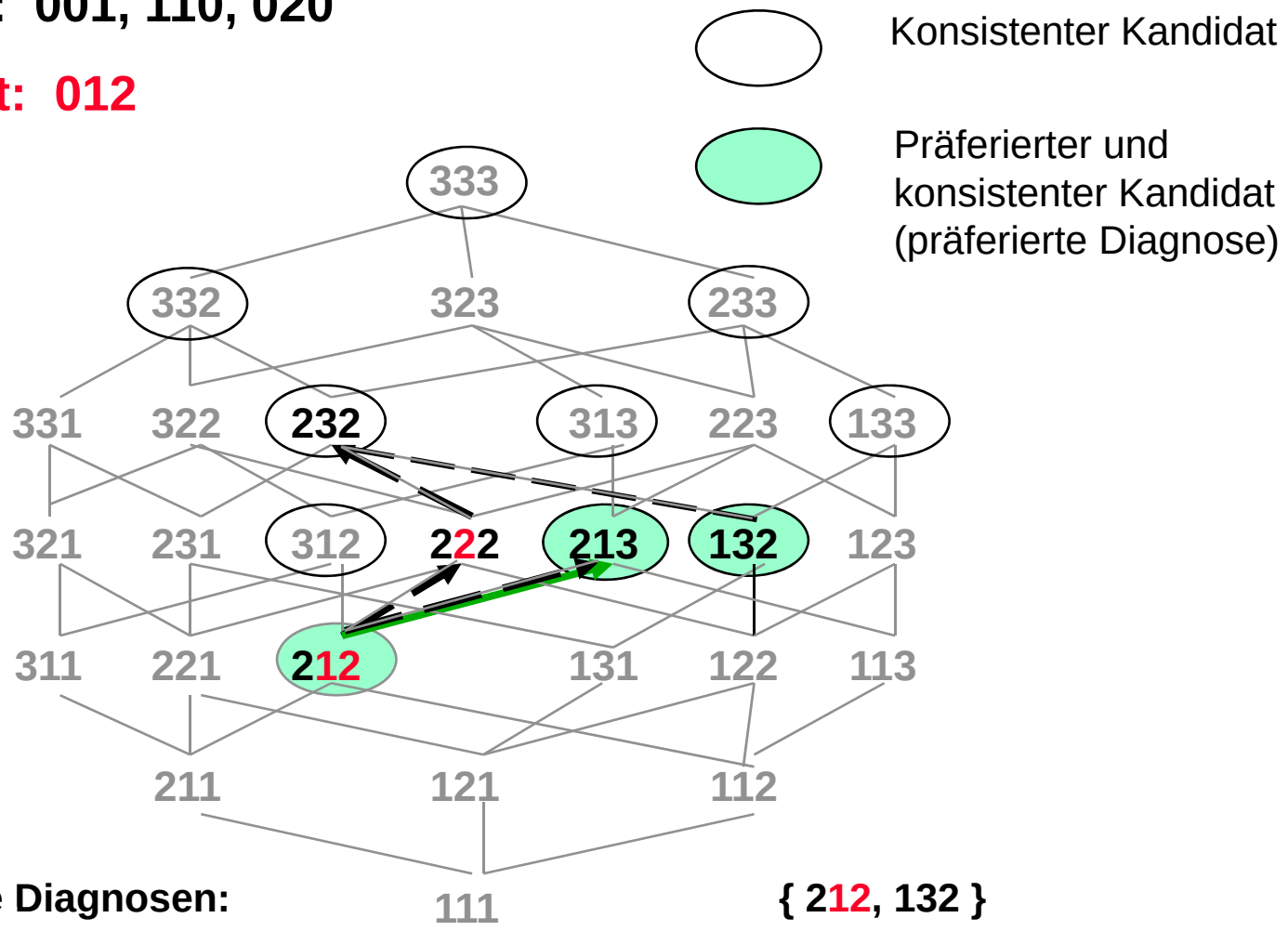
Neue Konflikte: 001, 110, 020



MDS: Kandidatenaktualisierung

Alte Konflikte: 001, 110, 020

Neuer Konflikt: 012



Bisherige präferierte Diagnosen: **{ 212, 132 }**
Neue präferierte Diagnosen (Stufe 1): **{ 222, 213, 132 }**
Neue präferierte Diagnosen (Stufe 2): **{ 213, 132 }**

MDS: Kandidatenaktualisierung

Aktionen bei Entdeckung eines neuen Konflikts:

1) Konsistenzcheck aller präferierten Diagnosen
2) Entfernen aller jetzt als inkonsistent erkannten Kandidaten
3) Bilden der Präferenznachfolger jedes eben entfernten Kandidaten
4) Aufnehmen der Präferenznachfolger, die folgende Bedingung erfüllen:
   - Der Nachfolger ist nicht von einer anderen noch als konsistent erkannten Diagnose präferiert.
   - Der Nachfolger ist selbst konsistent.

MDS: Kandidatenaktualisierung

Aktionen bei Entdeckung eines neuen Konflikts:

3) Bilden der Präferenznachfolger jedes eben entfernten Kandidaten:

- Wenn C der Konflikt ist, der in der alten Diagnose enthalten ist, dann Bilden nur der Nachfolger, die den Verhaltensmodus **genau einer Komponente** ändern, **die in C enthalten** ist

  (Bildung nur von direkten Nachfolgern, Nachfolgebildung bzgl. Konflikt C)

> ***Anmerkung: Auf diese Weise geht keine Diagnose verloren:***
>
> ***Satz:*** Jede Nachfolgediagnose, die C nicht enthält, ist Nachfolgediagnose eines direkten Nachfolgers, der C nicht enthält.

- Wenn einer dieser direkten Nachfolger einen Konflikt C' enthält, dann Bildung weiterer direkter Nachfolger bzgl. C'

MDS: Optimierungen der Kandidatengenerierung

1. Fokussierte Vorgehensweise:

- Ordne die präferierten und konsistenten Kandidaten nach Wahrscheinlichkeit
- Teile diese Kandidaten in 2 Mengen auf:
    - focus: Die besten k Kandidaten: Ihre vorhergesagten Werte werden nachfolgend vollständig weiterpropagiert, bis endgültige Konsistenz oder Inkonsistenz feststeht.
    - candidates: Die restlichen präferierten und konsistenten Kandidaten: Ihre vorhergesagten Werte werden vorerst nicht weiter propagiert

MDS: Optimierungen der Kandidatengenerierung

1. Fokussierte Vorgehensweise:

Aktionen bei Entdeckung eines neuen Konflikts:

1) Konsistenzcheck nur für die Fokusdiagnosen
2) Entfernen aller inkonsistenten Kandidaten aus dem Fokus
3) Bilden der direkten Präferenznachfolger jedes entfernten Fokuskandidaten bzgl. des jeweiligen Konflikts (siehe Detail von 3)
4) Einfügen dieser Nachfolger in candidates, sofern sie dort präferiert sind (anderenfalls werden sie verworfen)
5) Auffüllen von focus mit den wahrscheinlichsten Kandidaten aus candidates

MDS: Optimierungen der Kandidatengenerierung

2. Eliminierung irrelevanter Konflikte:

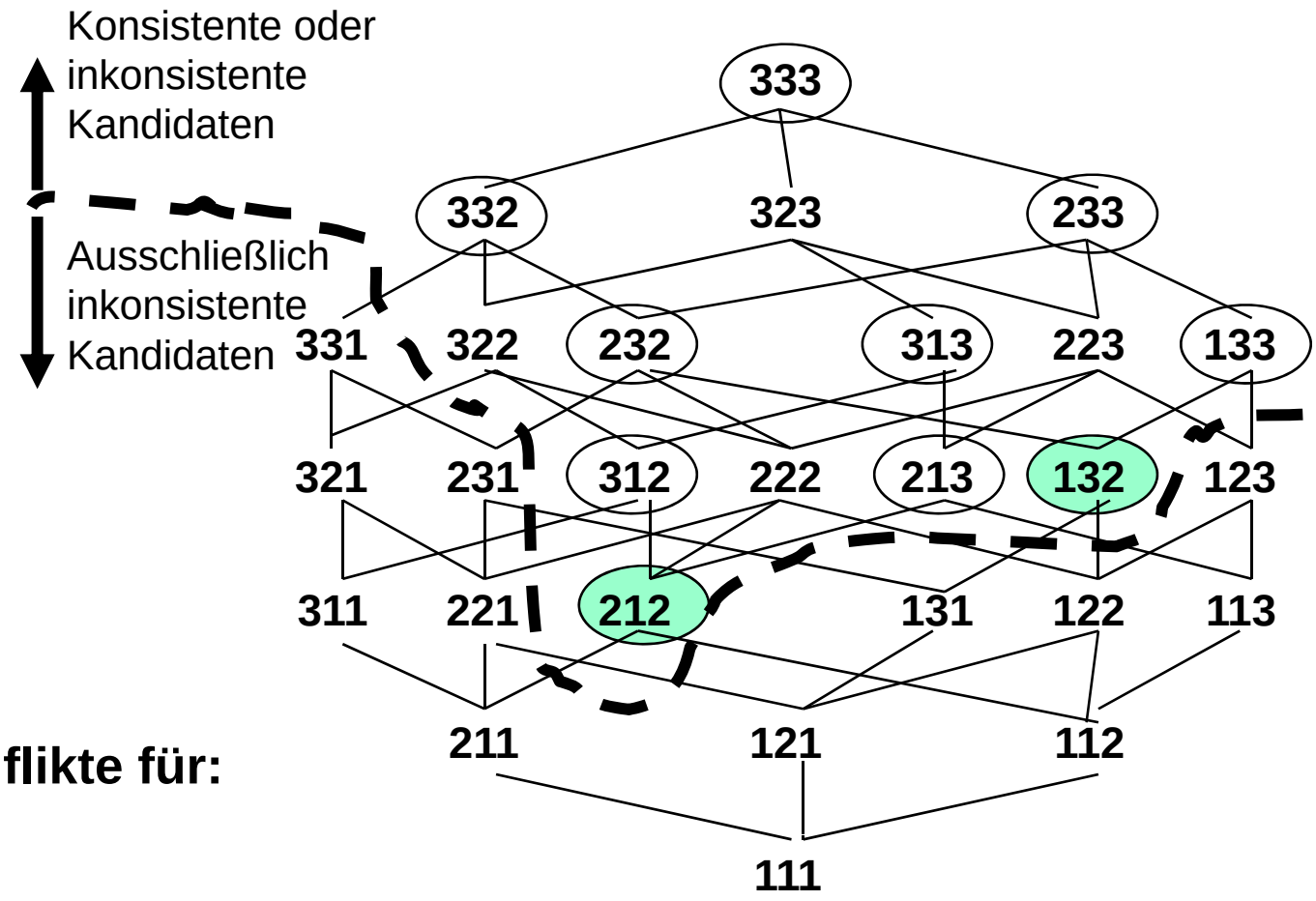
- Konflikte sind nur relevant, wenn sie einen Nachfolger der gegenwärtig präferierten Diagnosen zu Fall bringen könnten:
- Berücksichtige zum Kandidatentest nur die relevanten Konflikte !

Beispiel für relevante Konflikte:

Konflikt:	0 2 2	2 0 2	2 0 2
Kandidat:	3 1 2	2 1 1	1 1 3
Relevant ?			

MDS: Optimierungen der Kandidatengenerierung

2. Eliminierung irrelevanter Konflikte:



Konflikte:
{001, 110, 020}

Relevante Konflikte für:
- **212: 020**
- **132: keiner**

MDS: Optimierungen der Kandidatengenerierung

3. Effiziente Repräsentationsformen für:

- Kandidaten zum schnellen Test der Präferenzbeziehung
- Konflikte zum Testen der Relevanz für die gegenwärtigen Präferenzdiagnosen

Das Daimler-Produkt MDS enthält weitere Optimierungen zur Beschleunigung der Kandidatengenerierung

MDS: Konfliktgenerierung

Die Kandidatengenerierung löst folgende Aufgabe:

- **Gegeben eine Menge von Konflikten: Finde die wahrscheinlichsten präferierten Diagnosen zu diesen Konflikten.**
- **Damit reduziert sich das Diagnoseproblem auf folgende Aufgabe: Finde die Menge der Konflikte !**

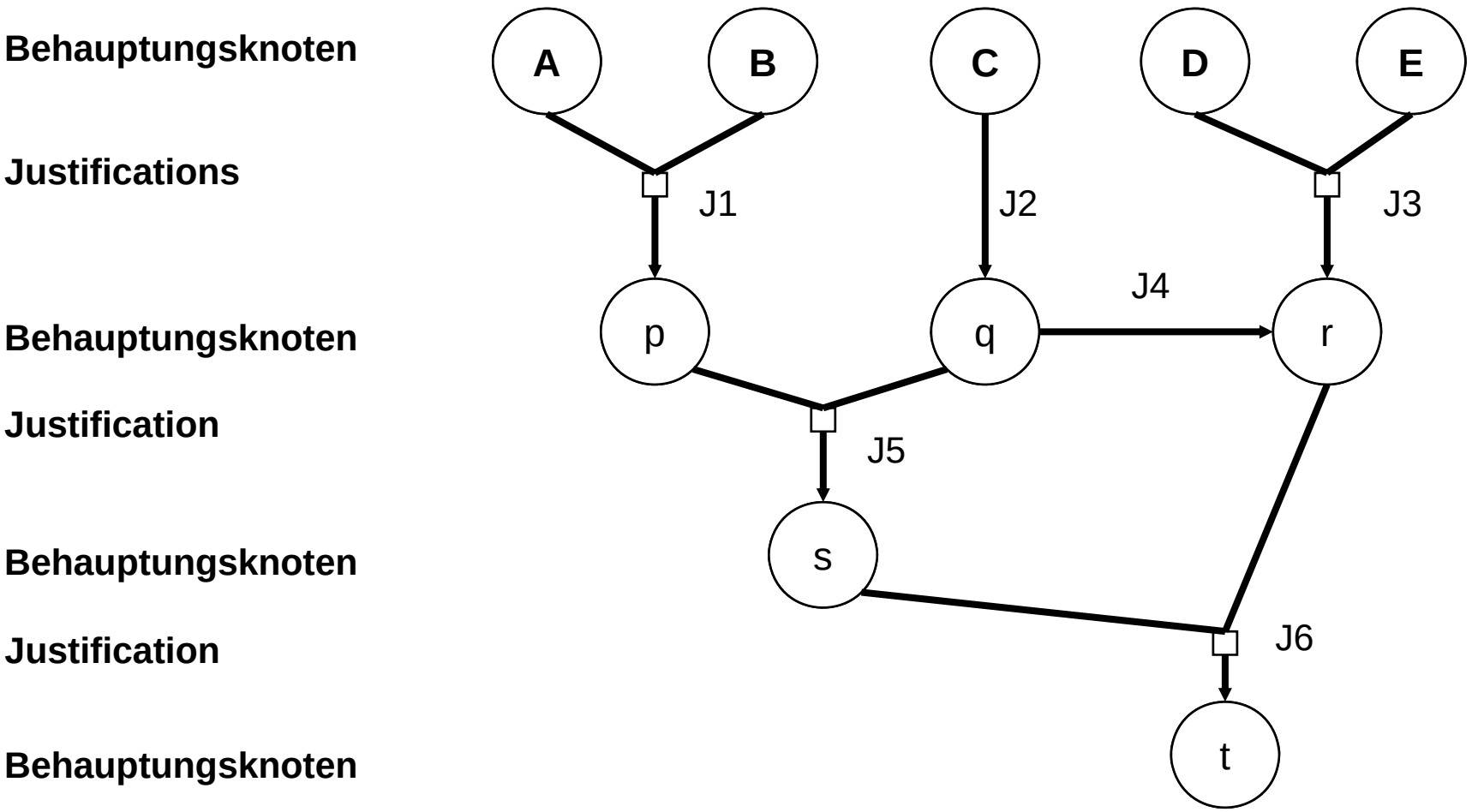
Was ist ein Konflikt ?

- **Zuweisung von genau einem Verhaltensmodus an einige Komponenten des Systems**
- **Ein Konflikt entspricht logisch einer Disjunktion von negativen Literalen**
- Zum Vergleich: Eine Diagnose entspricht einer Konjunktion von positiven Literalen

Wie entsteht ein Konflikt ?

- **durch Werte, die einander widersprechen**
- **Die sich widersprechenden Werte werden durch verschiedene Annahmen unterstützt.**
- **Dann muss eine dieser Annahmen falsch sein.**

TMS: Truth Maintenance System



Aus der Kombination von Behauptungen entsteht durch eine Justification eine neue Behauptung

TMS: Truth Maintenance System

Begriffswelt TMS:

Behauptungsknoten (propositional node):

steht für eine beliebige Aussage (kann wahr oder falsch sein)

Justification:

$A_1 \wedge A_2 \wedge ... \wedge A_n \Rightarrow C$ wobei $A_1, A_2, ..., A_n, C$ Behauptungsknoten sind

$A_1, A_2, ..., A_n$ heißen Antecedents (Vorgänger) der Justification

C heißt Conclusion (Folgerung) der Justification

Widerspruchsknoten ($\perp$):

Steht für eine Behauptung, die auf keinen Fall gilt

ATMS: Assumption-based Truth Maintenance System

Funktionalität eines allgemeinen TMS:

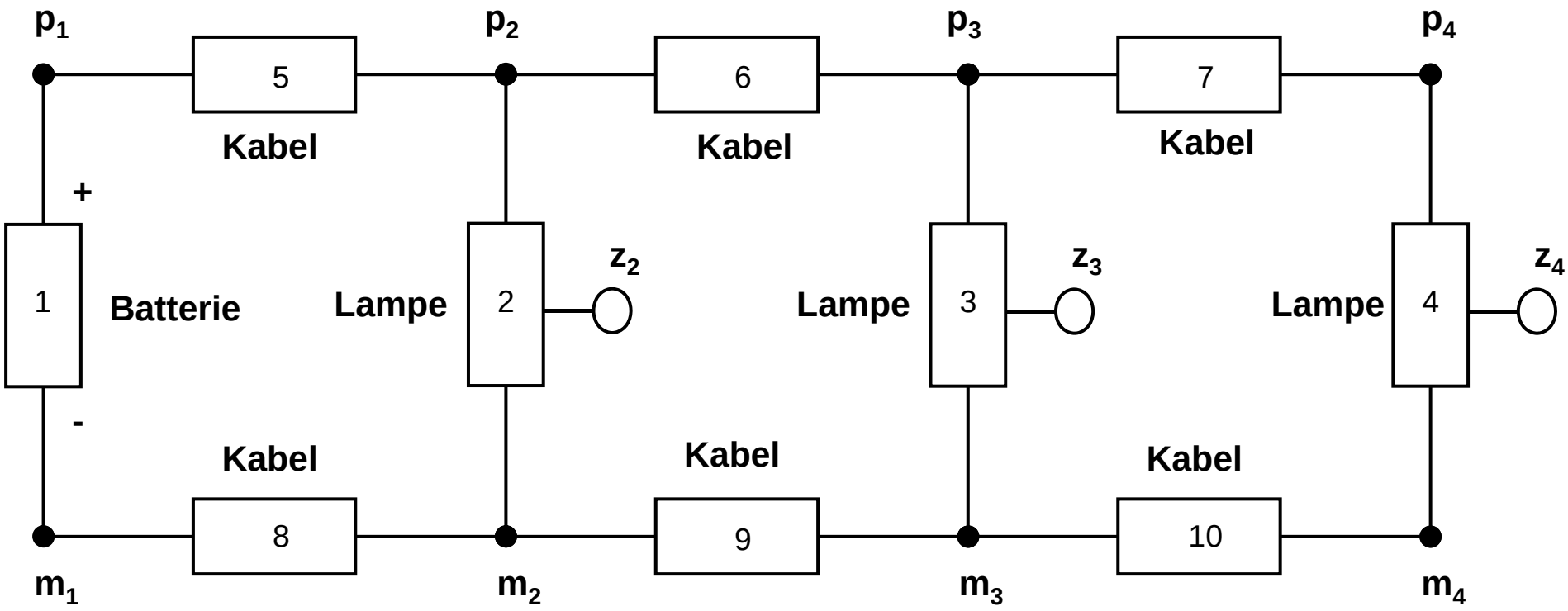
1) **Bestimmte Behauptungsknoten werden als wahr angenommen (beliefs).**
2) **Das TMS stellt durch Propagation dieser Annahmen über die Justifications fest, welche anderen Behauptungen dann auch gelten müssen.**
3) **Insbesondere wird durch Benutzung des Widerspruchsknotens festgestellt, ob die Kombination von bestimmten Annahmen in sich widersprüchlich ist.**

Zusätzliche Funktionalität eines ATMS:

- **Es wird gleichzeitig mit mehreren Kontexten gearbeitet: Ein Kontext ist die Menge verschiedener Annahmen, die gleichzeitig gelten sollen.**

1) **Die Behauptungen werden mit den Annahmekontexten versehen, unter denen sie gelten müssen.**
2) **Das ATMS stellt durch Propagation dieser Annahmekontexte über die Justifications fest, welche anderen Behauptungen dann auch gelten müssen.**
3) **Insbesondere wird durch Benutzung des Widerspruchsknotens festgestellt, welche Annahmekontexte in sich widersprüchlich sind.**

Beispiel für die Benutzung eines ATMS



Verhaltensmodi:

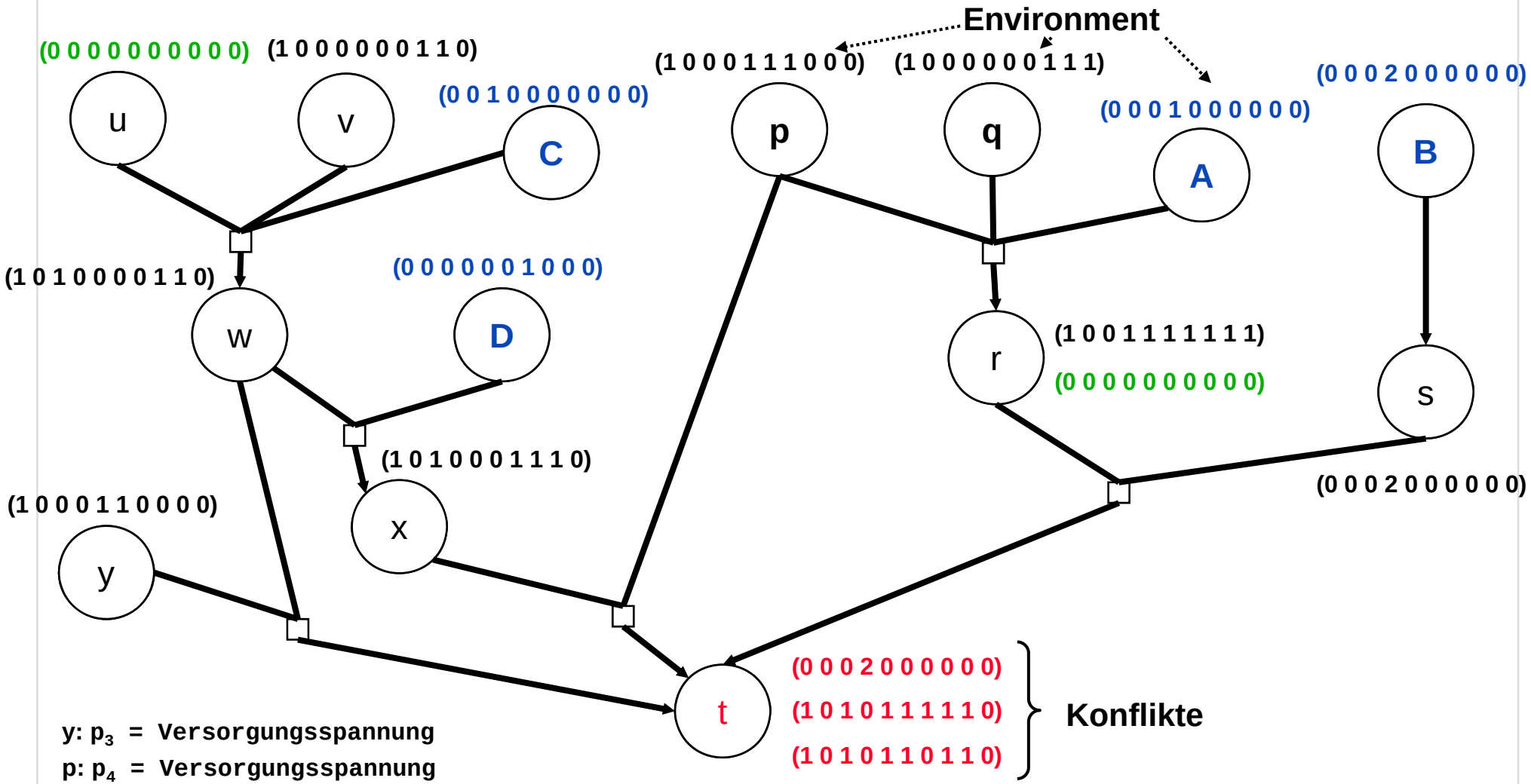
Modus 1 für alle Komponententypen: Normalverhalten

Modus 2 für alle Komponententypen: einziger Fehlermodus

Batterie: Modus 2 ⇒ (minus = `Masse`)

Lampe: Modus 2 ⇒ (z = `dunkel`)

Beispiel für die Benutzung eines ATMS



y: p_3 = Versorgungsspannung
p: p_4 = Versorgungsspannung

q: m_4 = Masse

A: Komponente 4 ist ok

r: z_4 = hell

B: Komponente 4 ist defekt

s: z_4 = dunkel

t: ⊥ (Widerspruch)

u: z_3 = dunkel

v: m_3 = Masse

C: Komponente 3 ist ok

w: p_3 = Masse

D: Komponente 7 ist ok

x: p_4 = Masse

ATMS: Assumption-based Truth Maintenance System

Begriffswelt ATMS:

Behauptungsknoten (propositional node):

Behauptungen werden unterschieden in normale Behauptungen und Annahmen (assumption node)

Environment:

Annahmekontext: *Konjunktion* von Annahmen, unter denen eine Behauptung gilt (wenn die Annahmen richtig sind)

Label:

Menge aller Annahmekontexte für einen Behauptungsknoten. Verschiedene Annahmekontexte müssen nicht gegeneinander konsistent sein (es gilt die *Disjunktion* der Environments).

Konflikt (nogood):

Environment des Labels des Widerspruchsknotens

ATMS: Assumption-based Truth Maintenance System

Anwendung eines ATMS für die modellbasierte Diagnose:

Behauptungsknoten (propositional nodes):

1) „Normale“ Knoten: Zuweisung eines konkreten Wertes an einer bestimmten Stelle (Variable) des Systems
2) Annahmeknoten: Zuweisung eines Verhaltensmodus an eine Komponente

Justification:

Anwendung einer Verhaltensregel auf konkrete Werte

Environment:

Gleichzeitige (Konjunktion) Zuweisung von Verhaltensmodi an Komponenten, unter der eine Behauptung gelten würde (Zuweisung muss nicht vollständig sein)

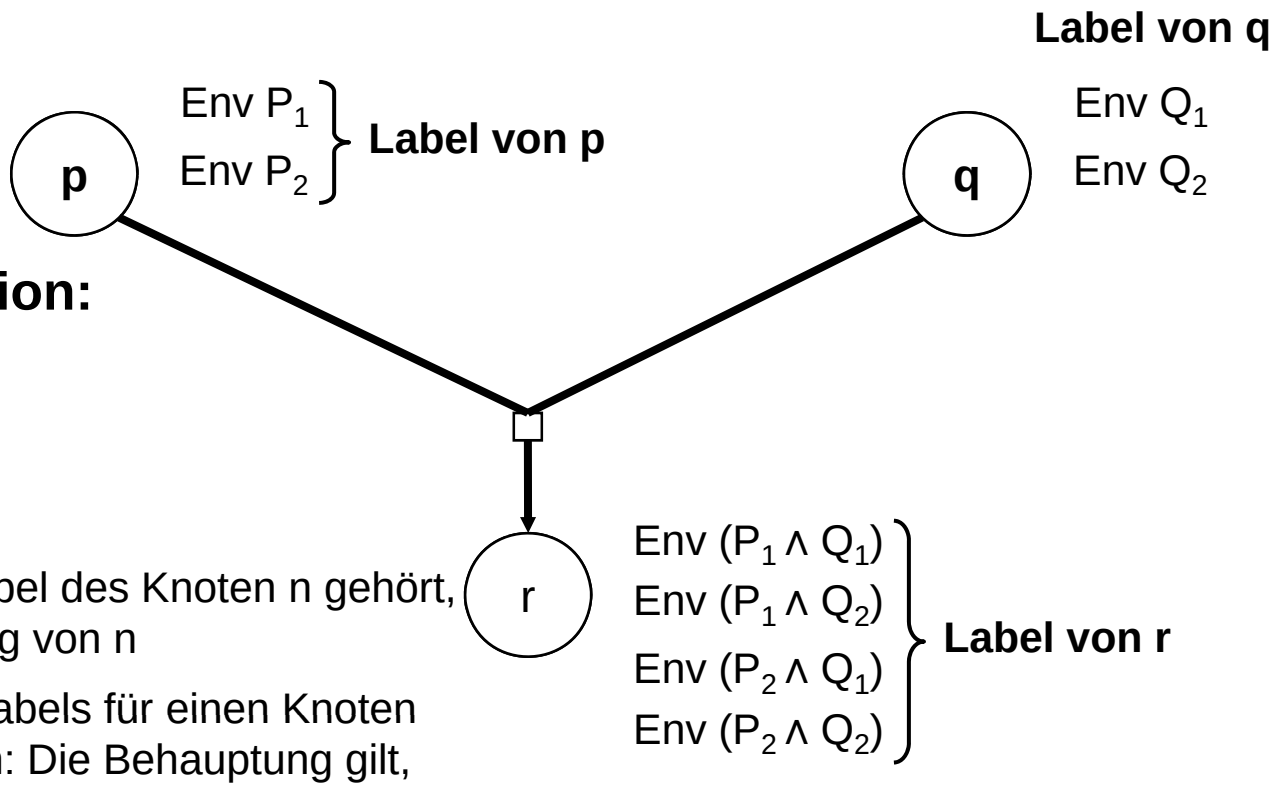
Bei Annahmeknoten: Zuweisung von einem Verhaltensmodus an genau eine Komponente

Konflikt (nogood):

Environment des Labels des Widerspruchsknotens: Zuweisung von Verhaltensmodi an Komponenten, deren Konjunktion nicht gelten kann (mindestens eine muss falsch sein)

Damit können Konflikte in der Notation und Bedeutung verwendet werden wie bei der Begriffswelt der GDE oben eingeführt.

Labelaktualisierung in einem ATMS



Bedeutung der Justification:

- r gilt, wenn p und q wahr sind (Konjunktion)

Bedeutung eines Labels:

- Wenn Environment e zum Label des Knoten n gehört, bedeutet das: e ⇒ Behauptung von n
- Mehrere Environments des Labels für einen Knoten entsprechen einer Disjunktion: Die Behauptung gilt, wenn eines der Environments wahr ist

Eliminierung überflüssiger Environments:

- Widersprüchliche Environments können weggelassen werden.
- Damit können auch alle Environments weggelassen werden, die Konflikte enthalten.
- Environments, aus denen andere Environments desselben Labels folgen, können weggelassen werden.

Bedienungsschnittstelle des ATMS

Eingabe vom Problemlöser:

- Annahmeknoten
- “Normale” Knoten
- Justifications zwischen den Knoten

Ausgabe an den Problemlöser:

- Menge der minimalen Konflikte

Das ATMS macht automatisch:

Das sind sehr viele Operationen !

- Erzeugung der Labels für die Annahmeknoten
- Aktualisierung der Labels aller Conclusions, von denen sich der Label eines Antecedents geändert hat
- Eliminierung aller überflüssigen Environments

Bedienungsschnittstelle des ATMS

Eingabe vom Problemlöser:

- Annahmeknoten
- “Normale” Knoten
- Justifications zwischen den Knoten

Ausgabe an den Problemlöser:

- Menge der minimalen Konflikte

Zusammenspiel mit dem Kandidatengenerierer:

- Bilde alle Annahmeknoten für die Fokusdiagnosen
- Wertpropagierung (Simulation):
  Berechne alle Werte, die sich aus den Annahmen der Fokusdiagnosen ergeben,
  bilde die entsprechenden Behauptungsknoten und Justifications,
  und gib diese in das ATMS ein.
- Entnimm dem ATMS die neuen Konflikte

Optimierung 1: Focusing ATMS

Eigenschaft, die verbessert werden soll:

- Das ATMS berechnet ***alle*** minimalen Konflikte, die zu einer beliebigen Kombination von Annahmen gelten.
- Die meisten davon sind für den Problemlöser gar nicht von Interesse.

Daher zusätzliche Eingabe vom Problemlöser:

- Diejenigen Environments, die von Interesse sind (Fokusenvironments)

Ausgabe:

- Menge der minimalen Konflikte, die in einem Fokusenvironment enthalten sind

Geänderte Funktionsweise des ATMS:

- Environments werden nur gebildet, wenn sie Teilmengen eines Fokusenvironments sind
- Bei Eingabe eines neuen Fokusenvironments werden automatisch alle Labels aktualisiert

Optimierung 2: Lazy ATMS

Eigenschaft, die verbessert werden soll:

- Das ATMS muss seine Labels häufig aktualisieren
- Viele Aktualisierungen werden nach außen gar nicht sichtbar, bevor sie von neuem überschrieben werden.

Geänderte Funktionsweise des ATMS:

- Labels werden erst berechnet, wenn nach ihnen explizit gefragt wird (in der Regel wird vor allem nach dem Label des Widerspruchsknotens gefragt)

In Daimler-MDS realisierte Variante:

- ATMS, das gleichzeitig fokussiert und lazy arbeitet

Beschreibung in der Dissertation von Mugur Tatar

Wertpropagierung und ATMS

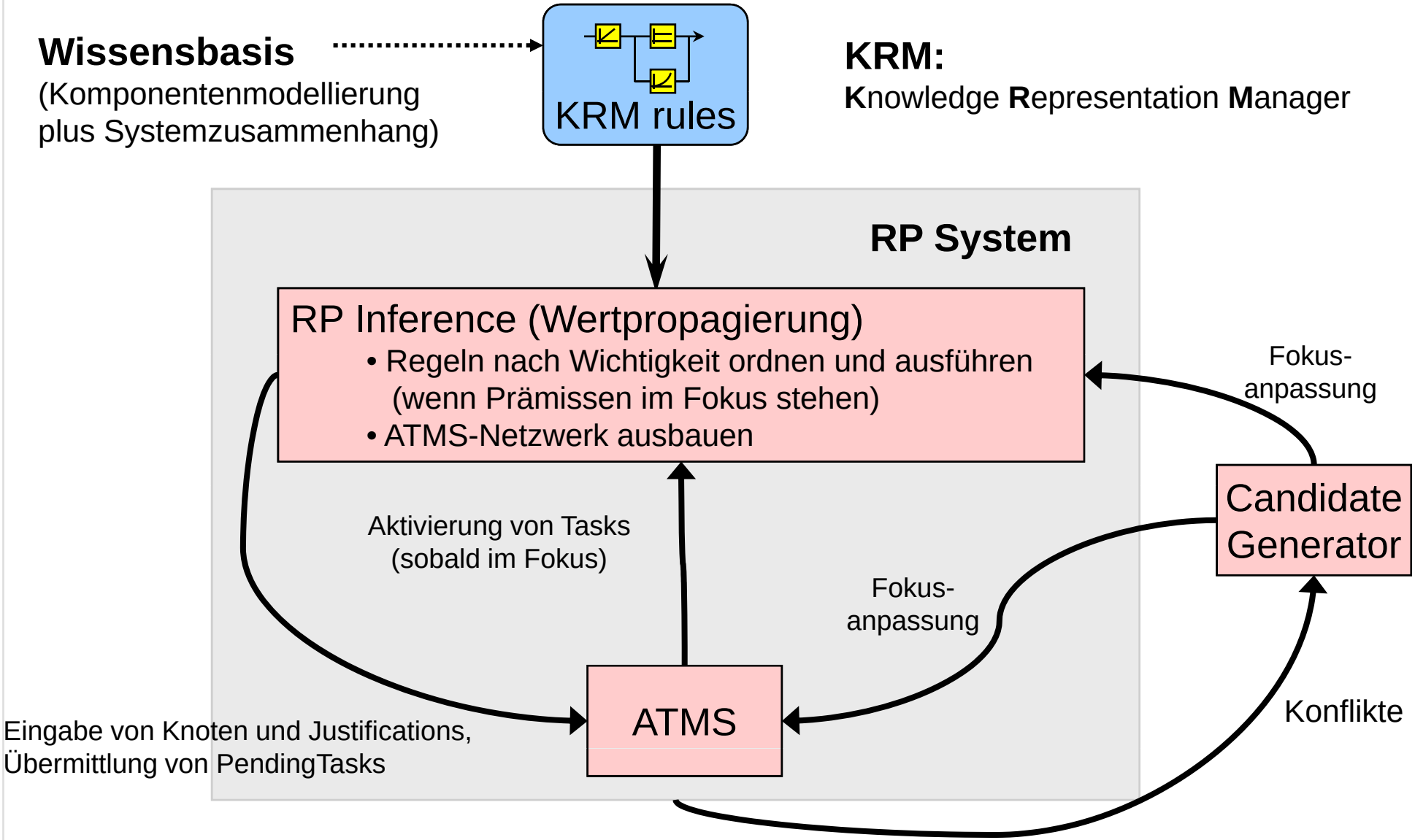
Was versteht man unter Propagierung allgemein ?

- Propagierung ist die Weiterleitung von Informationen über ein Netzwerk aus Kanten und Knoten.

Trennung von Wertpropagierung (RP) und ATMS:

- Das **ATMS** ist verantwortlich für die *Propagierung der Environments* in einem gegebenen Netzwerk von Wertabhängigkeiten.
- Die *Propagierung der Werte* wird von einem Rule Propagator (**RP**) vorgenommen, der die Justifications für konkrete Werte aus den allgemeinen Regeln für die Verhaltensmodi der Komponenten bildet und damit das vom ATMS benötigte Netzwerk von Wertabhängigkeiten.
- Der RP ist genauso faul wie sein ATMS:
  - Werte werden nur weiterpropagiert, wenn sie von Environments unterstützt werden, die im gegenwärtigen Fokus stehen.
  - Das ATMS teilt dem RP unaufgefordert mit, welche Werte neu von Fokusenvironments unterstützt werden und initiiert die **(Propagation) Tasks**

Wertpropagierung und ATMS



Wertpropagierung und ATMS

Was bringt die Trennung von Wertpropagierung und ATMS ?

→ Bessere Softwarearchitektur durch Modularisierung

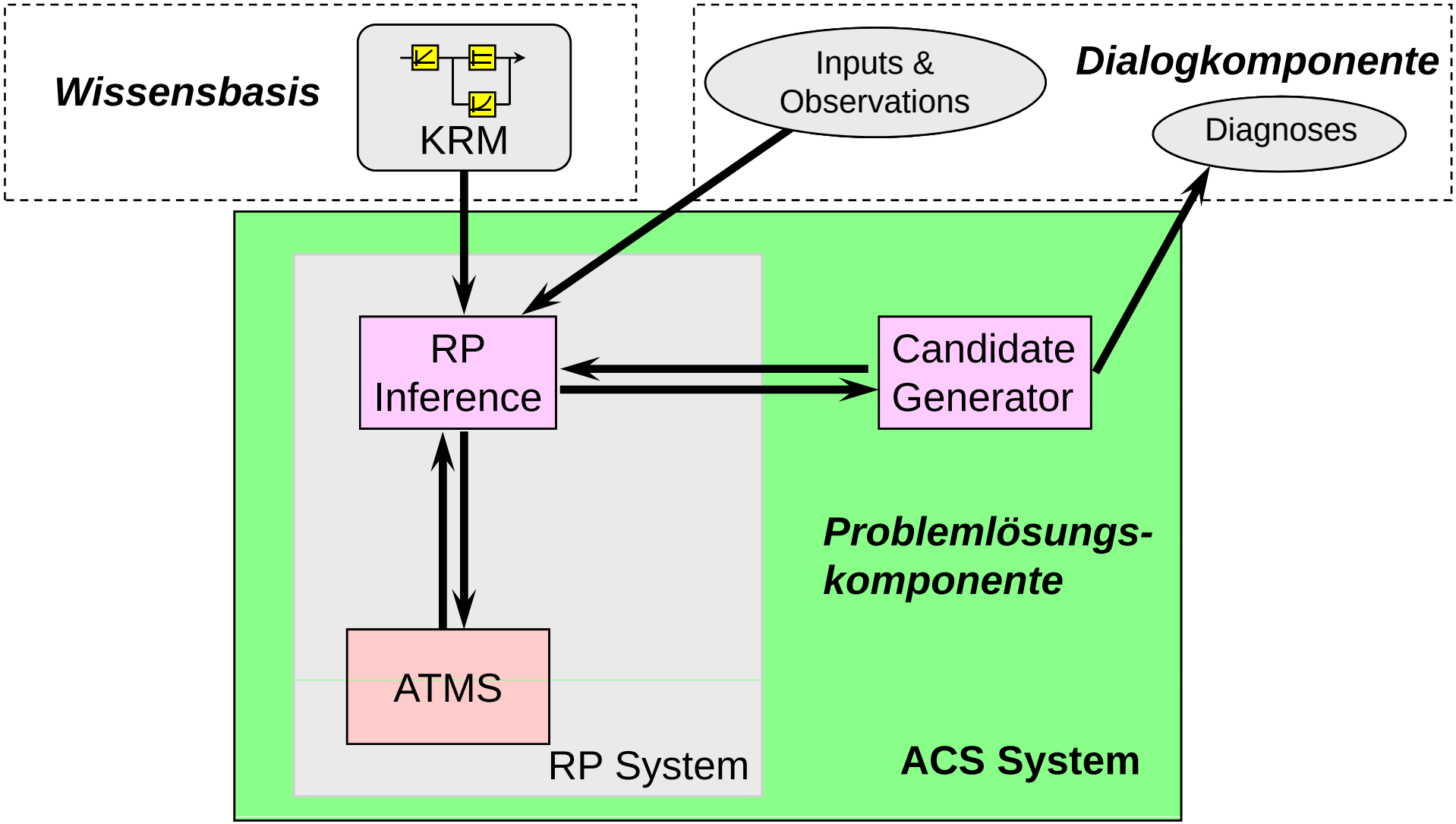
- **Werte** entstehen meistens aus Beobachtungen (Messungen) und gezielten Eingaben. Diese sind spärlich, daher gibt es **nicht viele** resultierende Werte.

- **Environments** entstehen aus Annahmen über Verhaltensmodi. Von diesen gibt es **sehr viele** (selbst bei Einfachfehlern mindestens so viele wie Komponenten).

- Daher werden Fokusenvironments häufiger revidiert, als neue Werte berechnet werden. Diese Revision kann dann als ein ATMS-internes Problem behandelt werden.

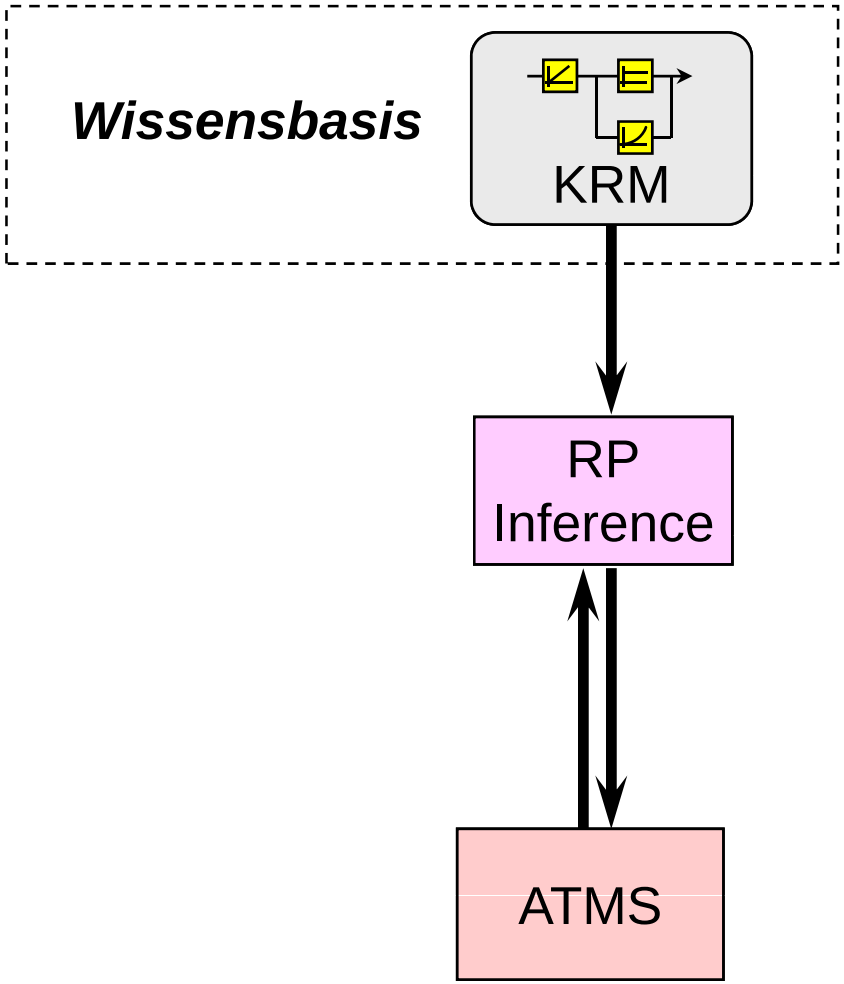
Anm.: Die Aufteilung in ein RP- und ATMS-Modul fördert enge Modulbindung und lose Modulkopplung

Zusammenspiel Kandidatengenerierer, RP und ATMS



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

Anforderung an die Wissensbasis



Was muss die Wissensbasis an die Inferenzkomponente liefern ?

- Regeln für die Wertzusammenhänge in den einzelnen Verhaltensmodi (*Komponentenmodellierung*)
- Kenntnis über die Wertdomänen: Wann gelten zwei Werte als widersprüchlich ?

Daimler-MDS löst diese Anforderungen durch das Anbieten einer Constraint-Sprache für die Komponentenmodellierung