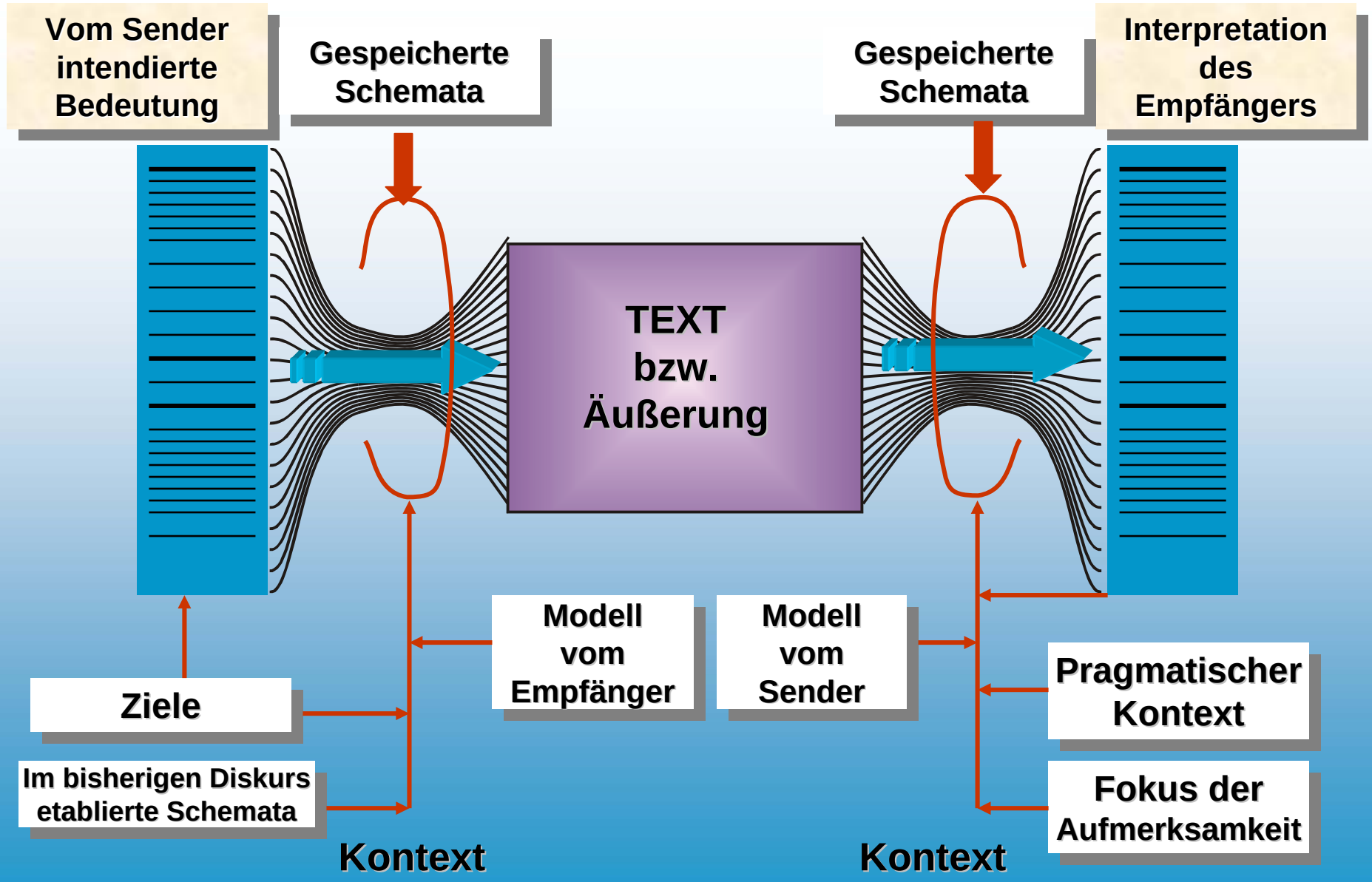
Semantik und Wissensrepräsentation

- Wissensrepräsentation
- deklarativ vs. prozedural
- Formen der Wissensrepräsentation


Vom Sender intendierte Bedeutung
Gespeicherte Schemata
Gespeicherte Schemata
Interpretation des Empfängers
TEXT bzw. Äußerung
Modell vom Empfänger
Modell vom Sender
Ziele
Im bisherigen Diskurs etablierte Schemata
Pragmatischer Kontext
Fokus der Aufmerksamkeit
Kontext
Kontext

Wissensrepräsentation

Wissensrepräsentation kann als die symbolische Darstellung von Wissen über einen Gegenstandsbereich definiert werden. Daraus leiten sich sofort zwei Fragen ab:

- was ist hier unter Wissen zu verstehen?
- in welcher Form kann dieses Wissen dargestellt werden?

Was ist unter Wissen zu verstehen?

Alltagssprachlich kann man von einer Person sagen, sie besitze Wissen über einen Sachverhalt, wenn folgende Bedingungen vorliegen:

1. der "Wissensträger" hält den Sachverhalt für wahr
2. der Sprecher hält diesen Sachverhalt ebenfalls für wahr
3. der "Wissensträger" kann den Sachverhalt beschreiben

Ist die 2. Bedingung nicht erfüllt, dann verwendet man im Deutschen anstelle von *wissen* die Verben *glauben* oder *meinen*. Ist die 3. Bedingung nicht erfüllt, verwenden wir *kennen* oder *können*.

Was ist unter Wissen zu verstehen?

Die 3. Bedingung verlangt auch, dass das Verb *wissen* zur Beschreibung von bewussten Zuständen verwendet wird, d.h. solche, die verbalisiert werden können.

In der Computerlinguistik und der KI-Forschung spricht man von Wissen jedoch auch dann, wenn die obigen Bedingungen 2. und 3. nicht erfüllt sind, d.h. auch dort, wo es darum geht, dass jemand etwas glaubt oder meint, oder etwas kennt oder kann.

Bei der Verwendung von können wird klar, dass es auch nicht nur um Sachverhalte sondern auch um Verfahren und Prozeduren geht (prozedurales Wissen).

Was ist unter Wissen zu verstehen?

Diese weitere Verwendung des Begriffs *Wissen* rührt u.a. daher, dass der deutsche Terminus *Wissensrepräsentation* eine Übertragung aus dem Englischen *Knowledge Representation* ist.

Nun lässt sich leicht zeigen, dass das englische Verb *know* und das deutsche *wissen* sich in ihrem Bedeutungsumfang nicht decken. Vielmehr umfasst *know* auch die Felder, die im deutschen mit *kennen* und *können* abgedeckt werden.

Deklaratives vs. prozedurales Wissen

Bei der Abgrenzung zwischen *wissen*, *kennen* und *können* ist schon die Unterscheidung zwischen sog. deklarativem und prozeduralem Wissen angeklungen.

In der KI wird entsprechend zwischen deklarativen und prozeduralen Formen der Wissensrepräsentation unterschieden.

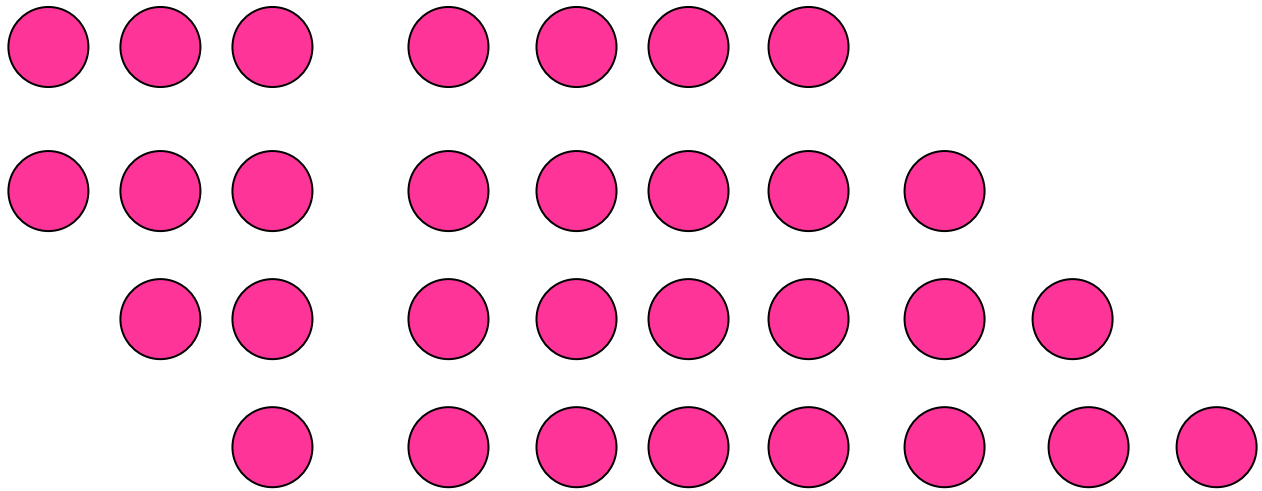
Dabei ist allerdings zu berücksichtigen, dass prozedurales Wissen und prozedurale Wissensrepräsentation sich nicht decken müssen. Auch deklaratives Wissen kann prozedural dargestellt werden und umgekehrt.

Deklarative Wissensrepräsentation

Deklarative Darstellungen von Wissensinhalten geben Beschreibungen von Sachverhalten, die keine Angaben über die Konstruktion und den Gebrauch von Wissen enthalten.
Beispiel:
"Die Summe aus 3 und 4 ist 7" oder als Formel: 3 + 4 = 7.

Prozedurale Wissensrepräsentation

Prozedurale Wissensdarstellungen beschreiben Verfahren zur Konstruktion, Verknüpfung und Anwendung von Wissen.
Beispiel:
Ein Verfahren zur Berechnung der Summe aus 3 und 4.



Kontrollwissen

Kontrollwissen nennt man Verfahren zur Steuerung des Einsatzes deklarativer und prozeduraler Wissensbeschreibungen. Kontrollwissen ist Metawissen.

Formen der Wissensrepräsentation

1. Wissensarten
   - Objekte
   - Ereignisse
   - Handlungswissen
   - Metawissen
2. Wissensrepräsentation
   - Formen allgemein
   - deklarative Wissensrepräsentation
   - prozedurale Wissensrepräsentation
3. Kontrollwissen

Wissensrepräsentation mit Begriffsgraphen

1. Begriffsgraphen
   - Begriffe (Konzepte)
   - Typen
   - Referenten
   - Begriffsrelationen
2. Kanonische Graphen
   - Kanonische Basis
   - Kanonische Formationsregeln
3. Typhierarchie
4. Typ-Definitionen
5. Schemata und Prototypen

Wissensarten

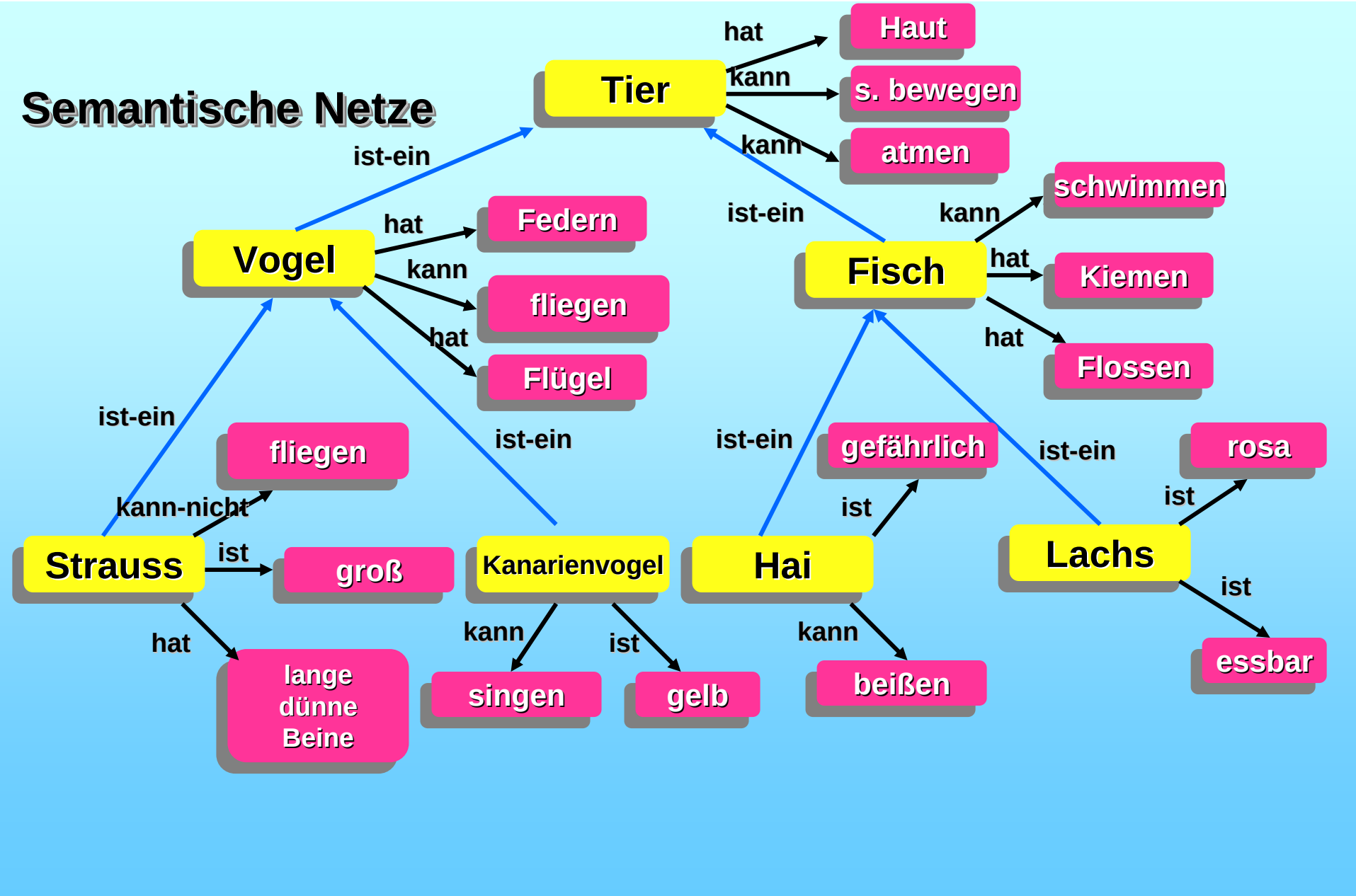
- **Objekte**
  Typischerweise betrachten wir Wissen als die Kenntnis von Fakten über Objekte in der Welt, die uns umgibt: Vögel haben Flügel. Schwalben sind Vögel. Schnee ist weiß. Wir müssen daher Objekte, Klassen oder Kategorien von Objekten, Beschreibungen von Objekten, und Beziehungen zwischen Objekten repräsentieren können.
- **Ereignisse**
  Wir haben auch Wissen über Vorgänge und Ereignisse in der Welt. Robert küsste Maria hinter dem Schuppen. Neben der Darstellung der Ereignisse selbst, muss ein Repräsentationsformalismus gegebenenfalls auch den zeitlichen Ablauf einer Ereignisfolge und die zwischen ihnen bestehenden Ursache-Wirkungs-Beziehungen erfassen können.

Wissensarten

- **Handlungswissen**
  Eine Fähigkeit wie z.B. Fahrradfahren erfordert neben dem Wissen über Objekte und Ereignisse auch Wissen darüber, wie bestimmte Handlungen auszuführen sind. Auch die meisten kognitiven Fertigkeiten wie z.B. die Bildung von Sätzen oder das Beweisen von Theoremen verlangen solches Handlungswissen.
- **Metawissen**
  Wir benutzen auch Wissen über unser Wissen, sog. Metawissen. Beispielsweise wissen wir etwas über den Umfang und die Herkunft unseres Wissens über einen spezifischen Gegenstand, über die Verlässlichkeit bestimmter Information, oder über die relative Wichtigkeit spezifischer Fakten über die Welt. Zum Metawissen gehört auch die Einschätzung unserer eigenen kognitiven Fähigkeiten sowie Wissen über Möglichkeiten des Wissenserwerbs.

Deklarative Formen der Wissensrepräsentation

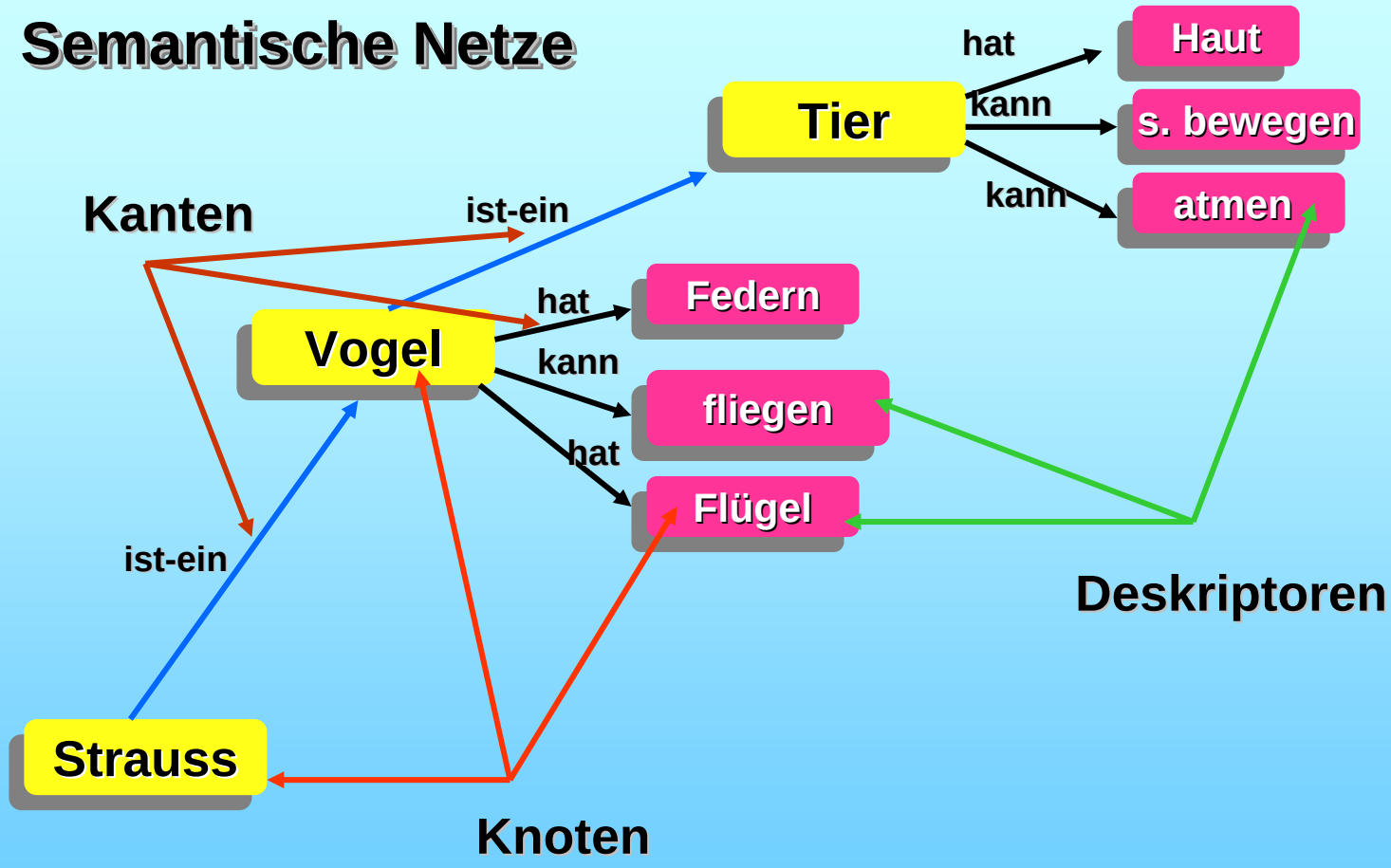
1. Semantische Netze
2. Objekt-Attribut-Wert-Tripel
3. Frames (Schemata, Scripts)
4. Produktionsregeln
5. Prädikatenlogik


Semantische Netze
Tier
hat
Haut
kann
s. bewegen
kann
atmen
ist-ein
Vogel
hat
Federn
kann
fliegen
hat
Flügel
ist-ein
Fisch
kann
schwimmen
hat
Kiemen
hat
Flossen
ist-ein
Strauss
kann-nicht
fliegen
ist
groß
hat
lange dünne Beine
ist-ein
Kanarienvogel
kann
singen
ist
gelb
ist-ein
Hai
ist
gefährlich
kann
beißen
ist-ein
Lachs
ist
rosa
ist
essbar

Semantisches Netz

Ein semantisches Netz ist ein gerichteter Graph aus einer Menge von Knoten, die Objekte (Begriffe oder Konzepte) repräsentieren, sowie einer Menge von gerichteten Kanten (engl. *arcs* oder *links*), die Beziehungen (Relationen) zwischen den Objekten darstellen. Normalerweise werden sowohl die Knoten als auch die Kanten (Verbindungen) mit Namen versehen.

Semantische Netze



Knoten

- Knoten werden benutzt, um Objekte und Deskriptoren zu repräsentieren.

Objekte

- Objekte können physische Gegenstände sein, die man sehen oder berühren kann. Objekte können auch gedankliche Elemente sein, wie z.B. Handlungen, Ereignisse oder abstrakte Kategorien.

Deskriptoren

- Deskriptoren liefern zusätzliche Informationen (Attribute, Eigenschaften) über Objekte.

Kanten (Verbindungen)

- Kanten repräsentieren Relationen, die Objekte und Deskriptoren miteinander verbinden. Einige häufige Verbindungen sind:

Ist-ein — **Damit wird häufig die Relation zwischen Klasse und Einzelfall repräsentiert: Waldi ist ein Dackel. Oft jedoch wird damit jedoch auch die Teilmengen-beziehung bzw. eine Subkategorie bezeichnet: Ein Dackel ist ein Hund. Diese beiden Verwendungen sollten jedoch besser auseinander gehalten werden. Beispielsweise könnte man die Relation zwischen Einzelfall und Klasse durch Element-von oder Instanz-von bezeichnen.**

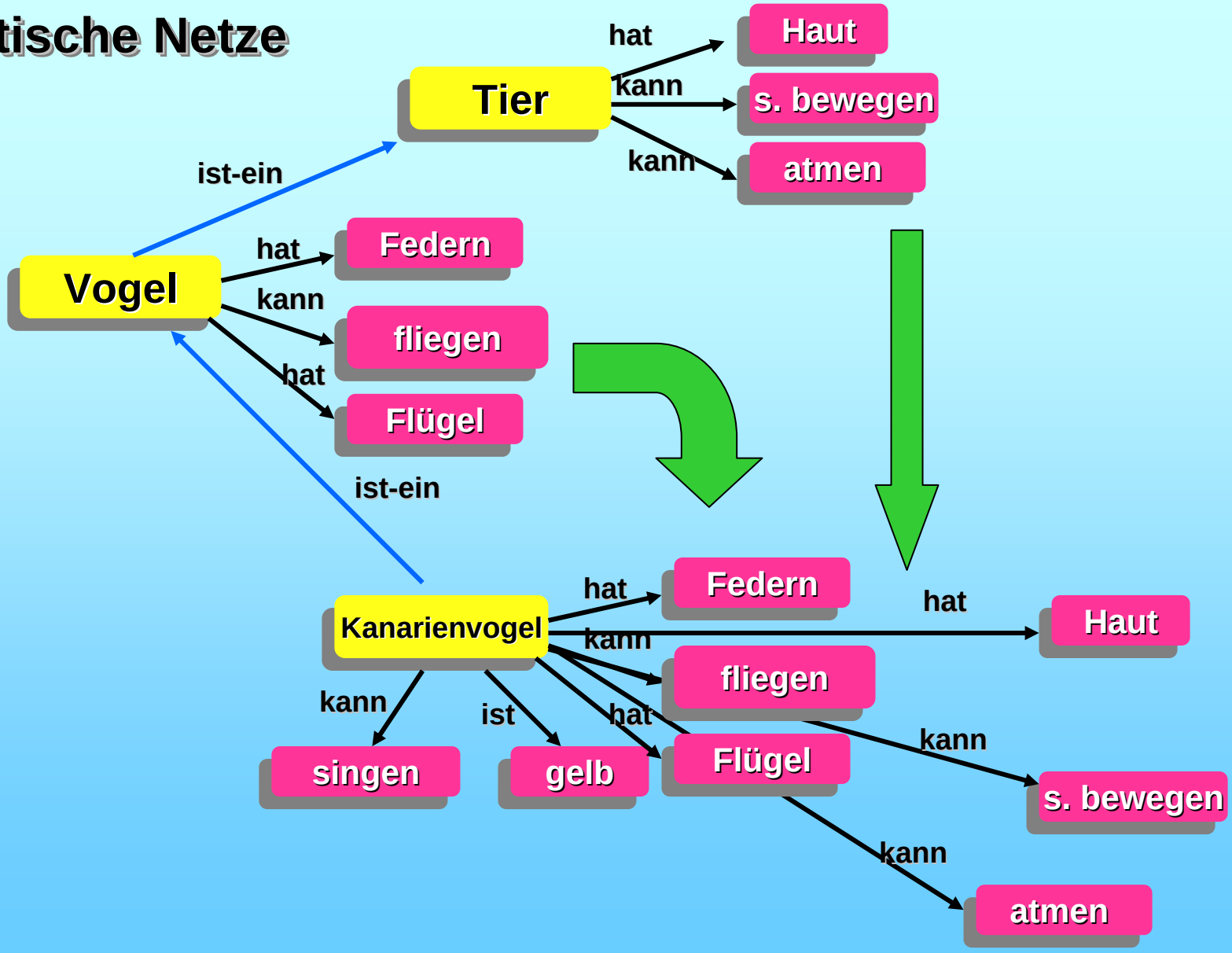
Hat — **Hat-Verbindungen bezeichnen Relationen zwischen Teilen und Teilelementen: Ein Hund hat einen Schwanz.**

Vererbung

- Dieser Begriff bezeichnet den Sachverhalt, dass ein Knoten die Charakteristika anderer Knoten, mit denen er verbunden ist "erben" kann. Die Vererbung von Eigenschaften ist eine Folge der ist-ein-Relation und bedeutet, dass alle Einzelfälle einer Klasse sämtliche Eigenschaften der übergeordneten Klassen, denen sie angehören, übernehmen.

```
instanz-von(waldi,dackel).
ist-ein(dackel,hund).
hat(hund,schwanz).
hat(X,Attribut):-ist-ein(X,Y),hat(Y, Attribut).
hat(X,Attribut):-instanz-von(X,Y),hat(Y,Attribut).
```

Semantische Netze



Objekt-Attribut-Wert-Tripel

Eine andere gebräuchliche Methode, um Wissensinhalte zu repräsentieren, ist die Darstellung als

Objekt-Attribut-Wert-Tripel

oder

O-A-W-Tripel (Assoziatives Tripel).

Es handelt sich dabei um einen Spezialfall der Darstellung durch semantische Netze.

Objekt-Attribut-Wert-Tripel

- Objekte sind entweder physische Entitäten oder begriffliche Einheiten.
- Attribute sind allgemeine Charakteristika oder Eigenschaften, die mit Objekten assoziiert werden. Größe, Form und Farbe sind typische Attribute von physischen Objekten.
- Der Wert eines Attributs kennzeichnet die spezifische Beschaffenheit (Ausprägung) eines Attributs in einer bestimmten Situation.

Objekt-Attribut-Wert-Tripel

Beispiele:

Objekt	Attribut	Wert
Apfel	Farbe	rot
Apfel	Herkunft	Israel
Apfel	Haltbarkeit	gut
Trauben	Farbe	blau
Trauben	Herkunft	Italien

Objekt-Attribut-Wert-Tripel

Beispiele:

Objekt	Attribut	Wert
Kindes	Kategorie	Nomen
Kindes	Genus	Neutrum
Kindes	Numerus	Singular
Kindes	Kasus	Genitiv
Kindes	Person	3

Objekt-Attribut-Wert-Tripel

Beispiele:

Objekt	Attribut	Wert
Kindes	Kategorie	Nomen
	Genus	Neutrum
	Numerus	Singular
	Kasus	Genitiv
	Person	3

Attribut-Wert-Paare

Beispiele:

Objekt	Attribut	Wert
Kindes	Kategorie	Nomen
	Genus	Neutrum
	Numerus	Singular
	Kasus	Genitiv
	Person	3

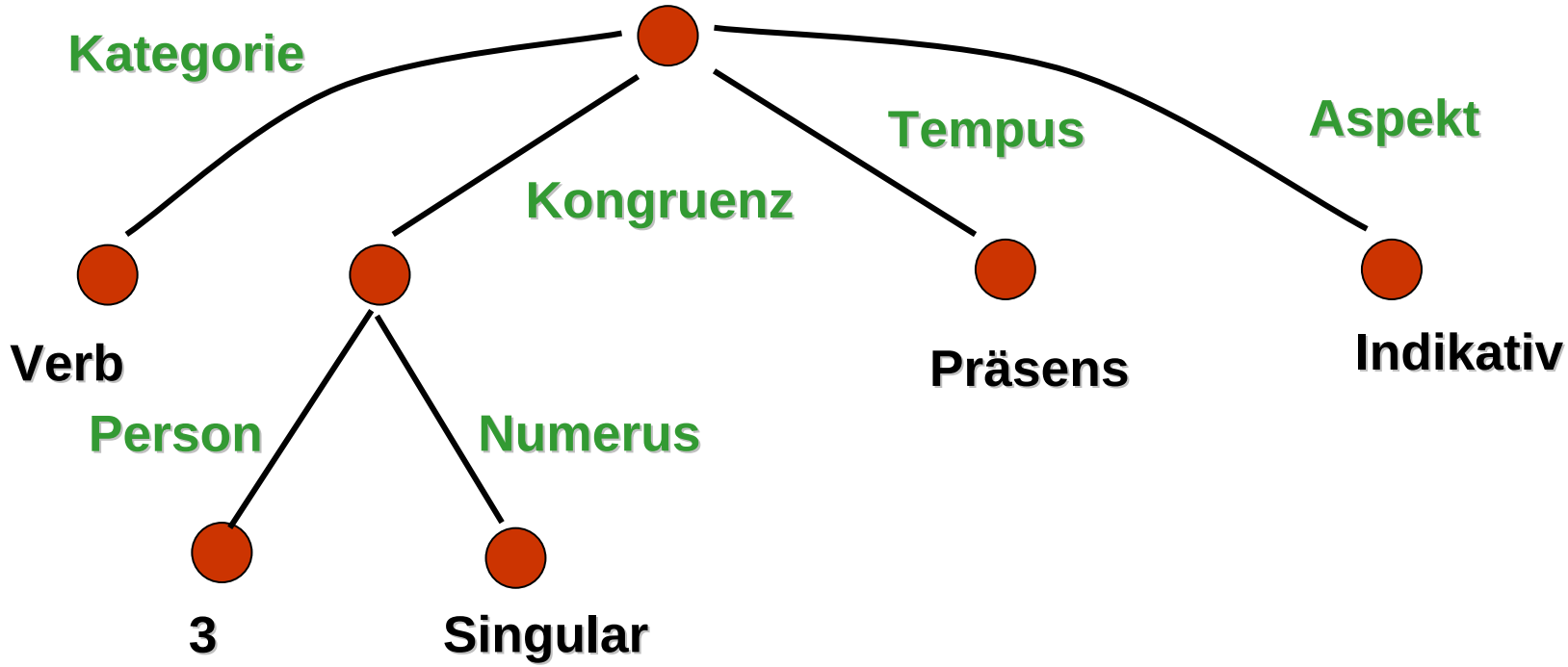
Attribut-Wert-Paare: Merkmalstrukturen

Beispiele:

singt

Kategorie	Verb
Kongruenz	Person: 3, Numerus: Singular
Tempus	Präsens
Modus	Indikativ

Attribut-Wert-Paare: Merkmalstrukturen



Frames

- When one encounters a new situation (or makes a substantial change in one's view of the present problem), one selects from memory a structure called a frame. This is a remembered framework to be adapted to fit reality by changing details as necessary.
- A frame is a data-structure for representing a stereotyped situation, like being in a certain kind of living room, or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed.
- We can think of a frame as a network of nodes and relations...

Frame

Ein Frame (Rahmen) ist eine Bündelung von Knoten und Attribut-Wert Paaren in einem semantischen Netz, die in ihrer Gesamtheit ein stereotypes Objekt, einen Akt, oder ein Ereignis beschreiben. Man kann einen Frame daher zunächst als eine Teilansicht in einem semantischen Netz auffassen.

Erweiterungen:

- Vorbelegungen (Default-Werte)
- "Prozedurale Anbindung" (procedural attachment)
- assoziierte Regelbündel

facets

slots

mammal	value	default	add	delete	calc
skin		fur			
birth	live				
legs		4			

facets

slots

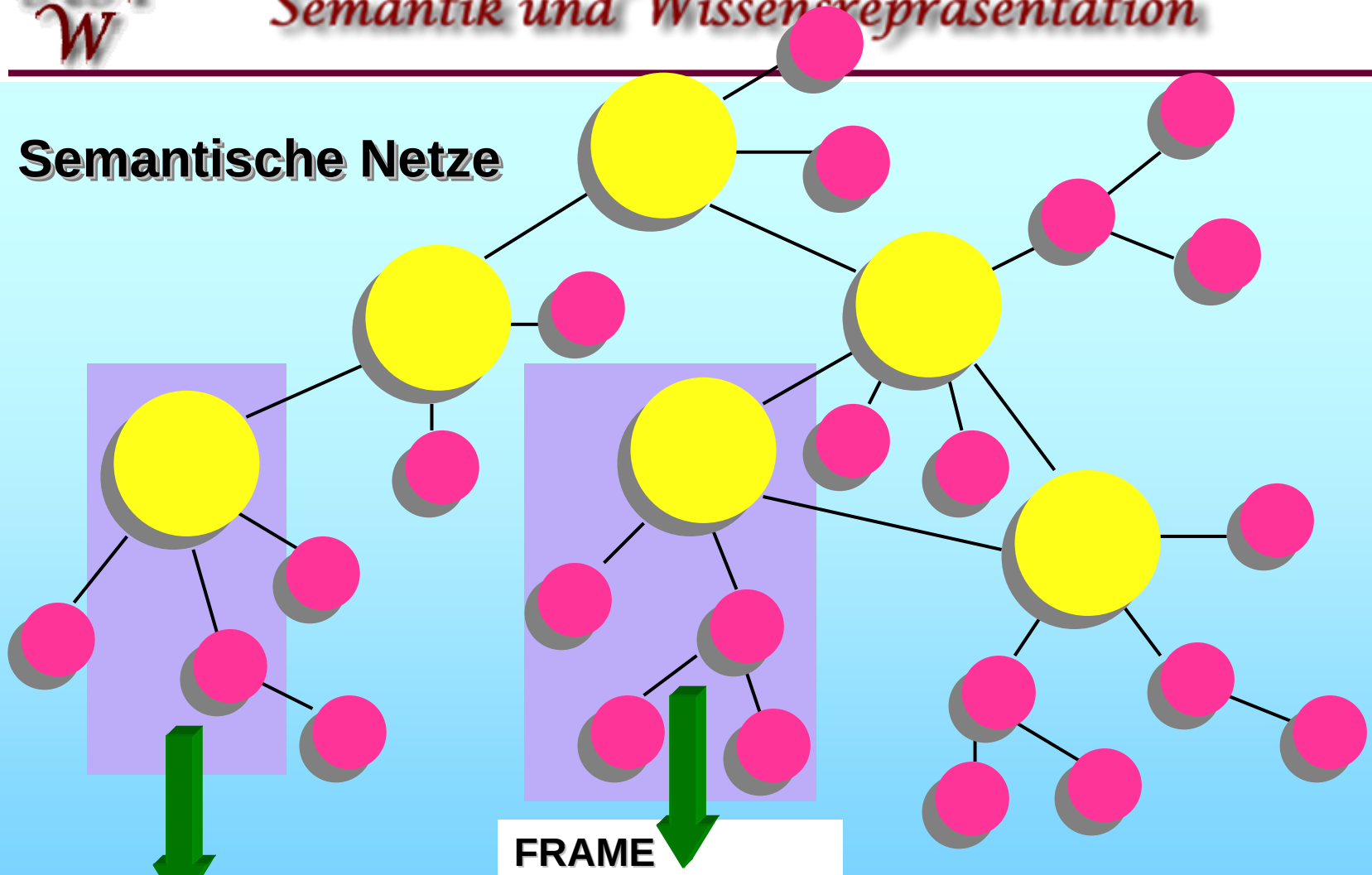
rabbit	value	default	add	delete	calc
ako	mammal				
ears		long			
moves	hops				

facets

slots

monkey	value	default	add	delete	calc
ako	mammal				
legs	2				
tail	curly				

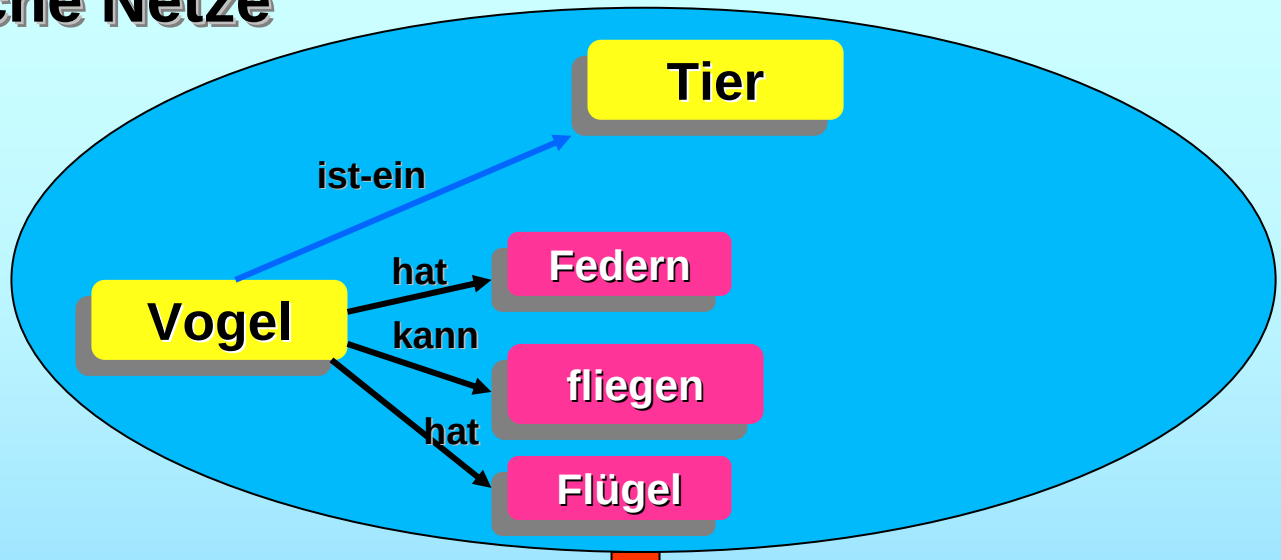
Semantische Netze



Objekt-Attribut-Wert-Tripel

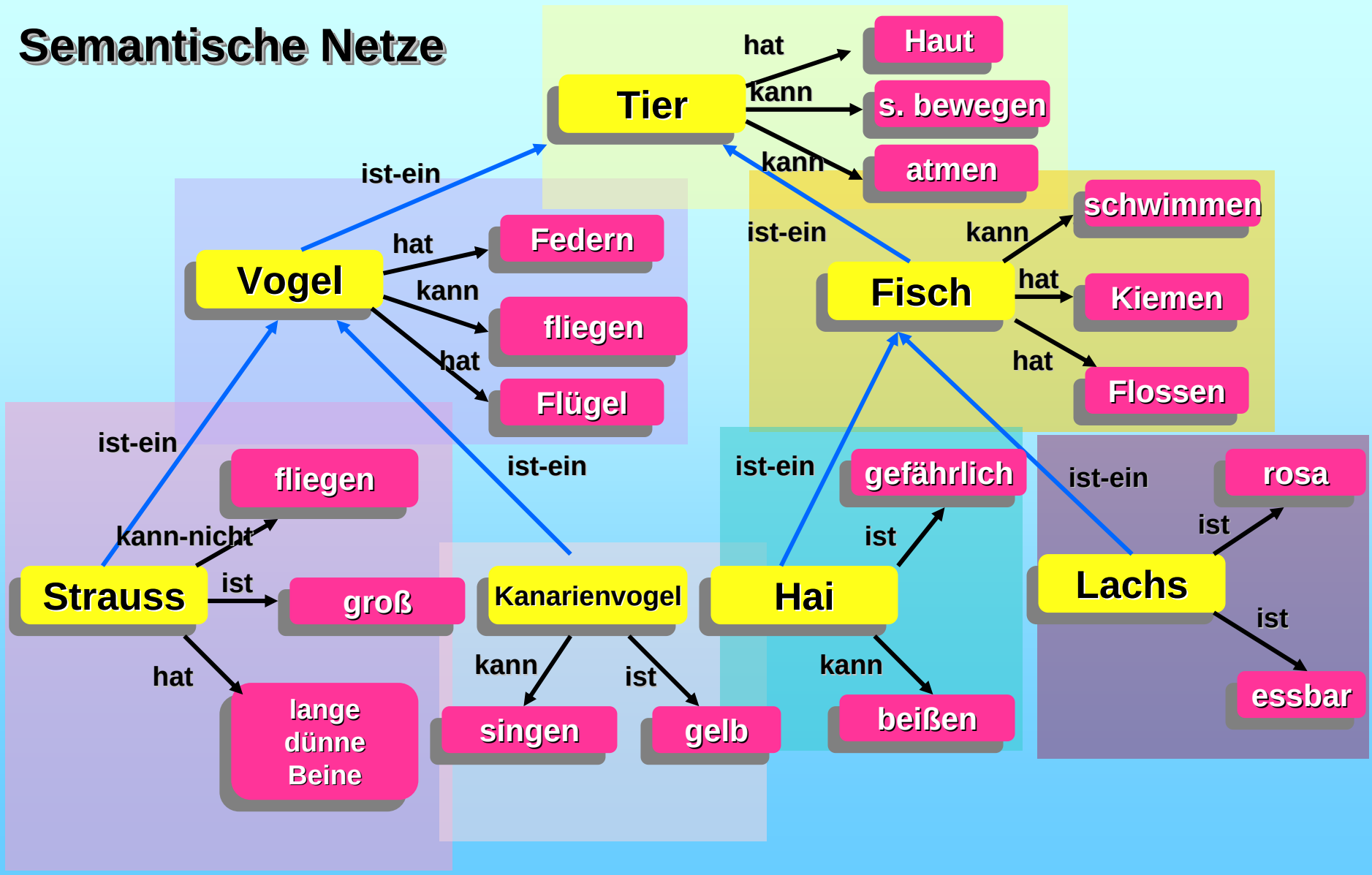
FRAME
Objekt: _____
Slot – Wert
Slot – Wert
Slot – Regel

Semantische Netze

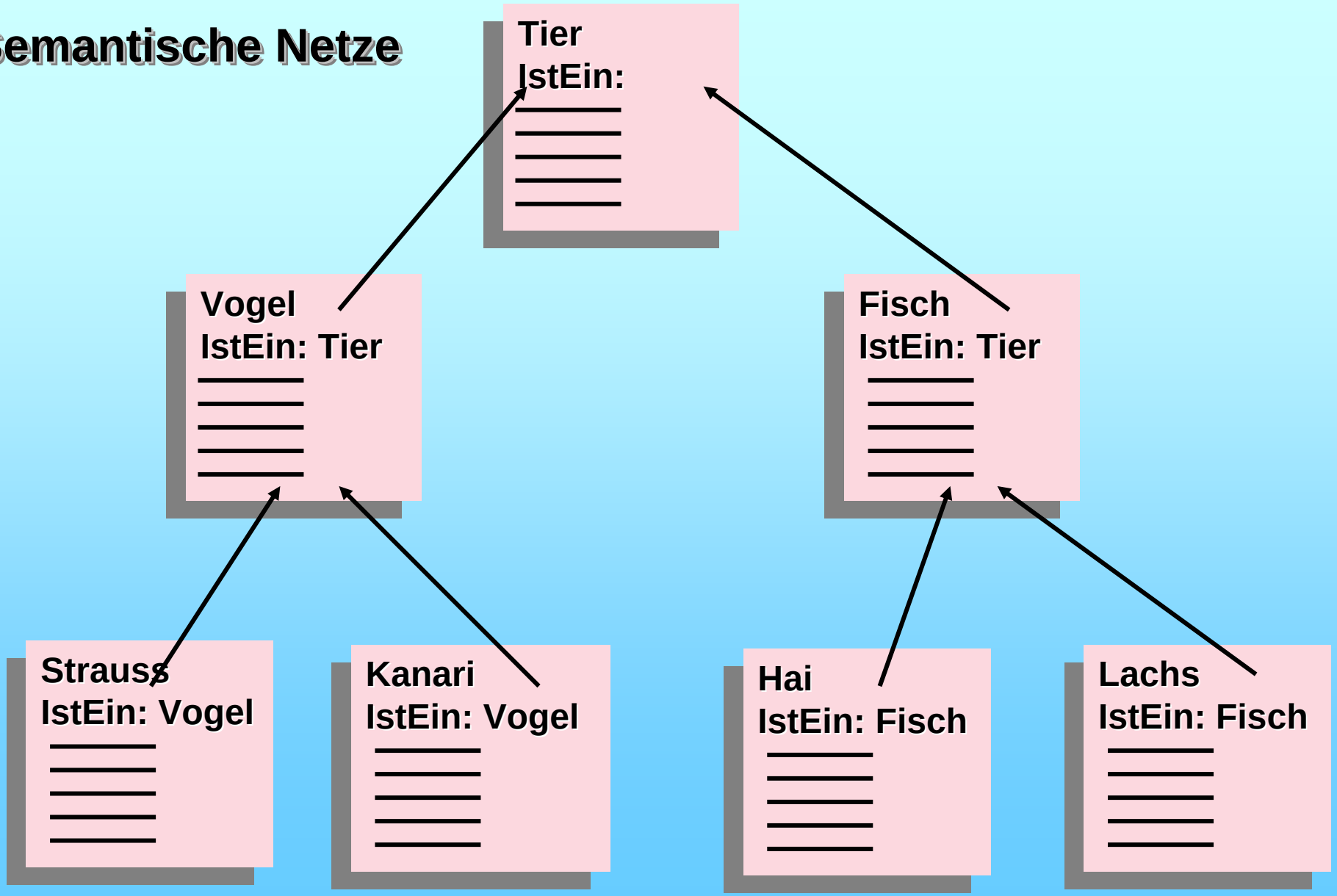


Vogel	
IstEin:	Tier
Hat:	Federn
Hat:	Flügel
Kann:	fliegen

Semantische Netze



Semantische Netze



Frames

Ereignis
IstEin:
Ort:
Tag:
Zeit:

Katastrophe
IstEin: Ereignis
Tote:
Verletzte:
Obdachlose:
Sachschaden:

Erdbeben
IstEin: Katastrophe
Verwerfung:
Stärke:

Überschwemmung
IstEin: Katastrophe
Wasserstand:
Gewässer:

Wirbelsturm
IstEin: Katastrophe
Name:
Windstärke:

Gesellsch. Ereignis
IstEin: Ereignis
Gastgeber:
Gäste:

Geburtstagsfeier
IstEin: Gesell. Ereignis
Alter:
Jubilar:

Hochzeit
IstEin: Gesell. Ereignis
Braut:
Brauteltern:
Bräutigam:
Eltern d. Br.:
Brautkleid:

Sportereignis
IstEin: Ereignis
Sportart:
Gewinner:
Ergebnis:

FRAMES

Erdbeben in Neurelien

- Heute ereignete sich in Neurelien ein schweres Erdbeben von einer Stärke von 8.5. Das Beben tötete 25 Personen. Es gab 523 Verletzte. Der Sachschaden beträgt DM 500.000.000. Der Präsident von Neurelien teilte mit, dass das hart getroffene Gebiet in der Nähe der Santa Anna Verwerfung schon seit Jahren eine Gefahrenzone gewesen sei.

Zusammenfassung (Muster)

- <Wert im Tag-Slot> ereignete sich in <Wert im Ort-Slot> ein Erdbeben. Es gab <Wert im Tote-Slot> Tote, <Wert im Verletzte-Slot> Verletzte, und einen Sachschaden in Höhe von DM <Wert im Sachschaden-Slot>. Die Stärke des Bebens betrug <Wert im Staerke-Slot> auf der Richter Skala, und die verursachende Verwerfung war <Wert im Verwerfung-Slot>.

FRAMES

Zusammenfassung (Instanziierung)

- Heute ereignete sich in Neurelien ein Erdbeben. Es gab 25 Tote, 523 Verletzte, und einen Sachschaden in Höhe von DM 500.000.000. Die Stärke des Bebens betrug 8.5 auf der Richter Skala, und die verursachende Verwerfung war Santa Anna.

Erdbeben-13 **IstEin: Erdbeben**	
Ort:	**Neurelien**
Tag:	**heute**
Tote:	**25**
Verletzte:	**523**
Sachschaden:	**500,000,000**
Stärke:	**8.5**
Verwerfung:	**Santa Ana**

Regelbasierte Systeme

Regelbasi eme

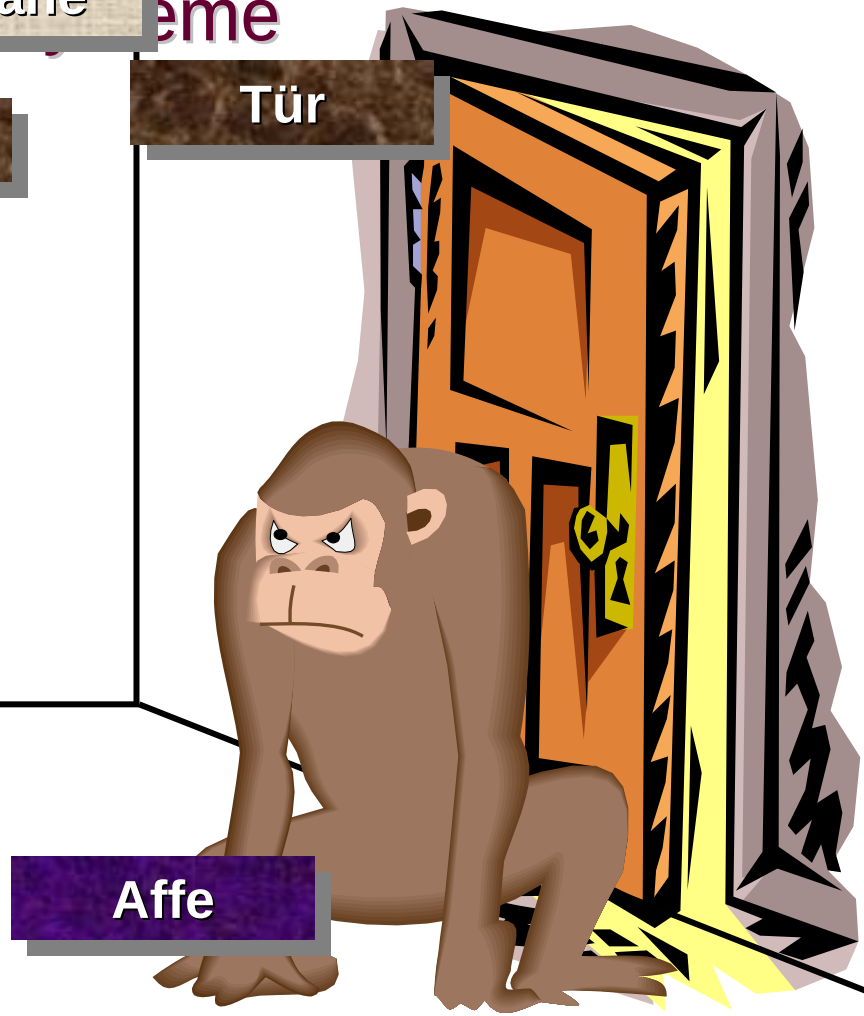
Fenster

Banane

Mitte

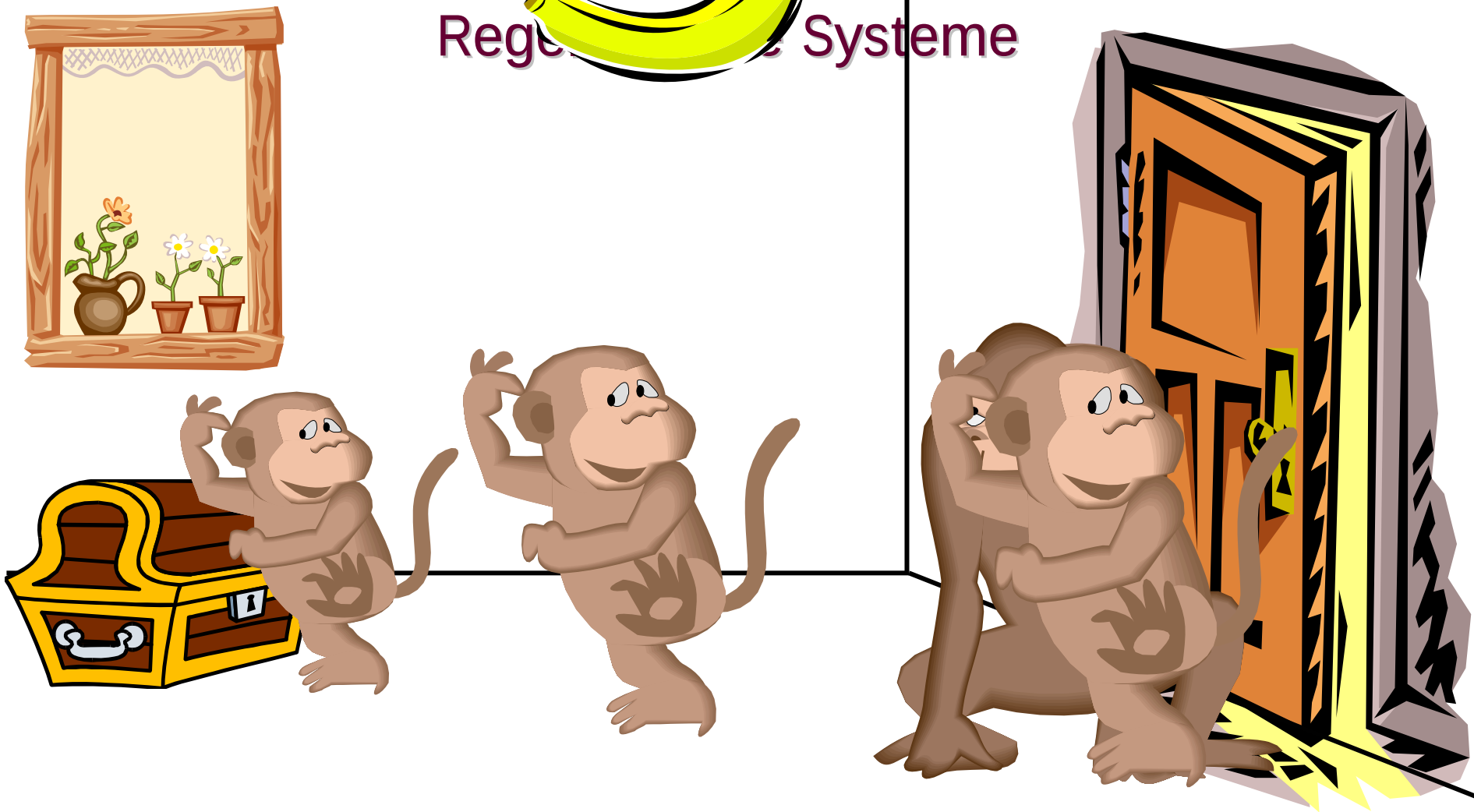
Tür

Kiste

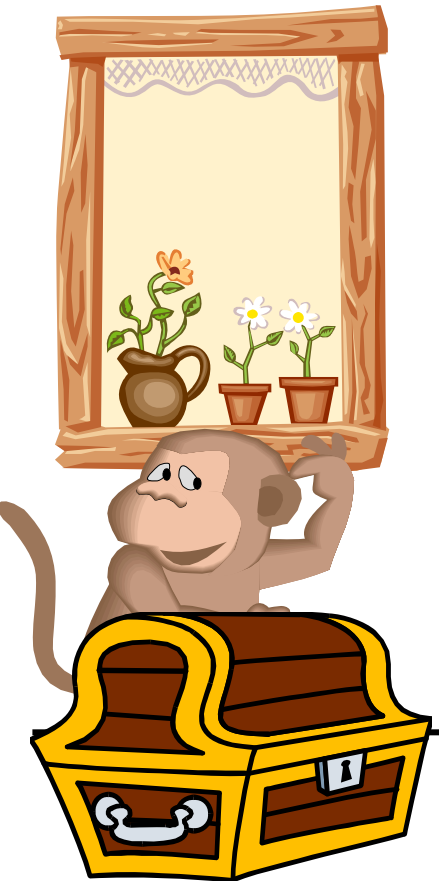


Affe

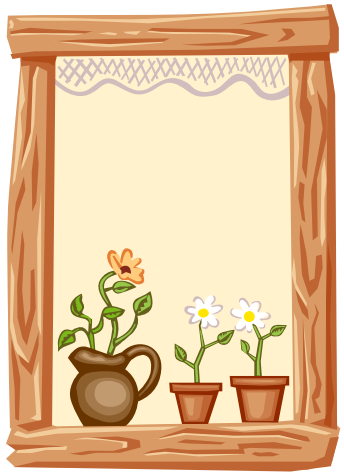
Regelbasierte Systeme

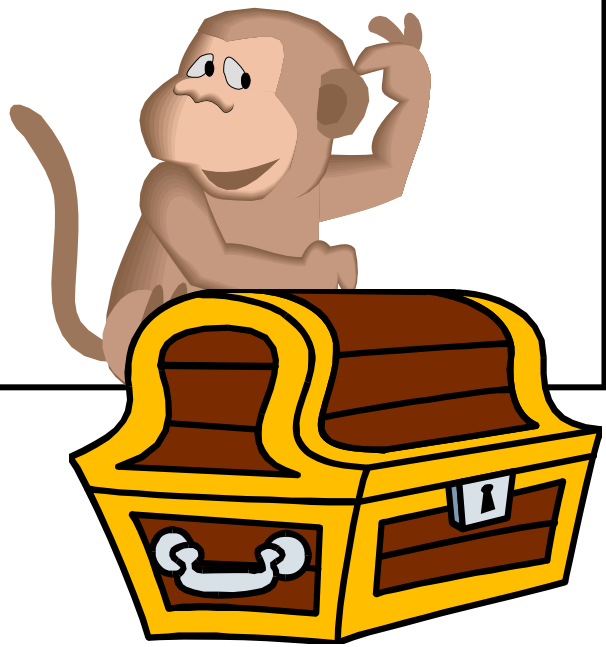


Regelbasierte Systeme

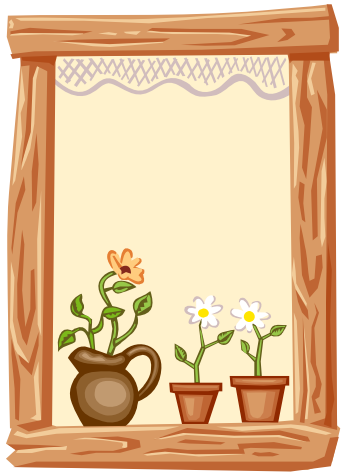


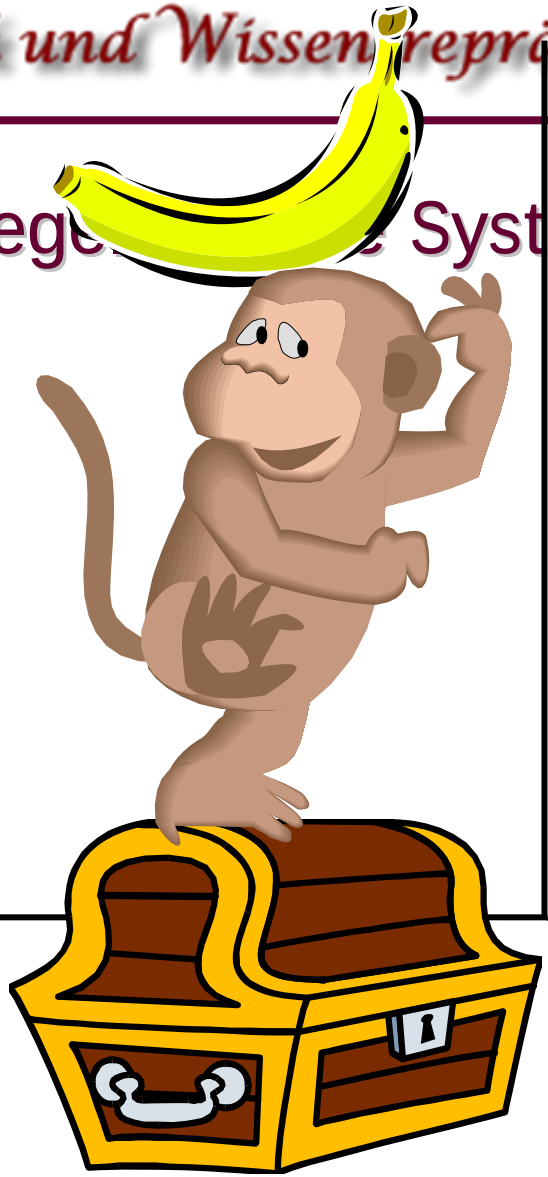
Regelbasierte Systeme



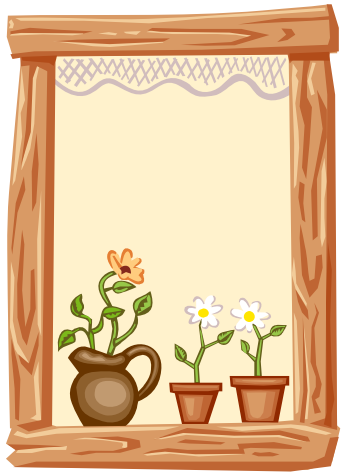


Regelbasierte Systeme





Regelbasierte Systeme



Regelbasierte Systeme

Wenn	Dann
<BEDINGUNG>	<AKTION>
Bedingungen können in Form von Objekt-Attribut-Wert-Tripeln oder Attribut-Wert-Paaren notiert sein	

Regelbasierte Systeme

Wenn	Dann
Affe hat Banane	Affe kann Banane essen

Regelbasierte Systeme

Wenn	Dann
Affe hat Banane nicht und Kiste.Ort = Banane.Ort und Affe steht auf Kiste	Affe ergreift Banane

Regelbasierte Systeme

Wenn	Dann
Affe steht auf Boden **und** **Kiste.Ort = Banane.Ort** **und** **Affe.Ort = Kiste.Ort**	**Affe klettert auf Kiste**

Regelbasierte Systeme

Wenn	Dann
Kiste.Ort =/= Banane.Ort und Affe.Ort = Kiste.Ort	Affe schiebt Kiste von Kiste.Ort zu Banane.Ort

Regelbasierte Systeme

Wenn	Dann
Affe.Ort =/= Kiste.Ort	Affe geht von Affe.Ort zu Kiste.Ort

Regelbasi eme

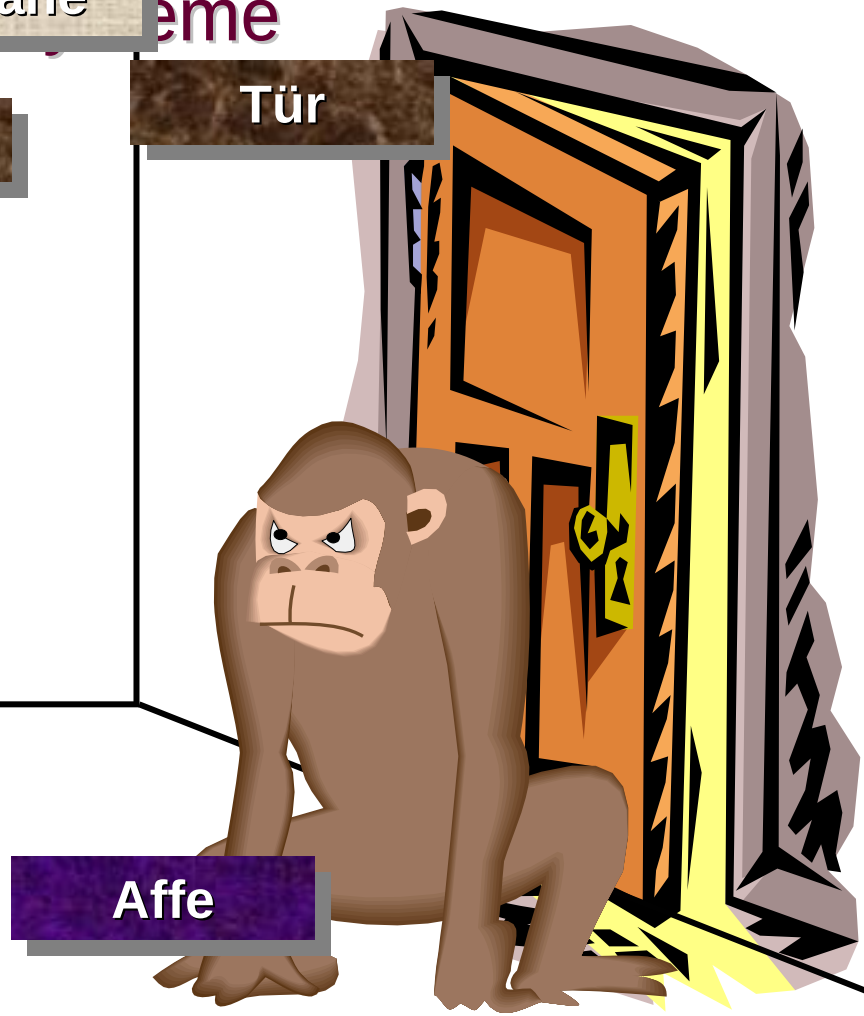
Fenster

Banane

Mitte

Tür

Kiste



Affe

Regelbasierte Systeme

Objekt	Attribut	Wert	Wertebereich
Kiste	Ort	_	Tür, Mitte, Fenster
Banane	Ort	_	Tür, Mitte, Fenster
Affe	Ort	_	Tür, Mitte, Fenster
	hat Banane	_	ja, nein
	Position	_	auf Kiste, auf Boden

Regelbasiert Systeme: Umsetzung in Prolog

zustand(<Affe>, <Banane>, <Kiste>).

<Affe>:= affe(<Ort>, <Position>, <hatBanane>).

<Banane>:= banane(<Ort>).

<Kiste>:= kiste(<Ort>).

<Ort>:= {tuer, fenster, mitte}.

<Position>:= {auf_boden, auf_kiste}.

<hatBanane>:= {ja, nein}.

Regelbasiert Systeme: Umsetzung in Prolog

```
zustand(affe(tuer, auf_boden, nein), banane(mitte), kiste(fenster)).

kann_essen(Zustand):-
        Zustand=
                zustand(
                        affe(_, _, ja),
                        banane(_),
                        kiste(_)
                        ).

kann_essen(zustand(
                        affe(_, _, ja),
                        banane(_),
                        kiste(_)
                        )).
```

Regelbasiert Systeme: Umsetzung in Prolog

```
kann_essen(Z1):-
        zustandsaenderung(Z1, _Aktion, Z2),
        kann_essen(Z2).

zustandsaenderung(
        zustand(affe(O, auf_kiste, nein), banane(O), kiste(O)),
        ergreift,
        zustand(affe(O, auf_kiste, ja), banane(O), kiste(O)).
```

Regelbasierte Systeme: Umsetzung in Prolog

```
zustandsaenderung(
        zustand(affe(O, auf_boden, B), banane(O), kiste(O)),
        klettert,
        zustand(affe(O, auf_kiste, B), banane(O), kiste(O)).

zustandsaenderung(
        zustand(affe(O1, auf_boden, B), banane(O2), kiste(O1)),
        schiebt(O1,O2),
        zustand(affe(O2, auf_boden, B), banane(O2), kiste(O2)).

zustandsaenderung(
        zustand(affe(O1, auf_boden, B), banane(O), kiste(O2)),
        geht(O1,O2),
        zustand(affe(O2, auf_boden, B), banane(O), kiste(O2)).
```

Regelbasierte Systeme: Umsetzung in Prolog

```
start:-
    anfangszustand(Affe,Banane,Kiste),
    kann_essen(zustand(Affe,Banane,Kiste),Aktionen),
    schreibe_pfad(Aktionen).

anfangszustand(affe(Ort1,Kiste,nein),banane(Ort2),kiste(Ort3)):-
    write('Wo befindet sich der Affe? (tuer, mitte, fenster) '),read(Ort1),nl,
    write('Wo befindet sich die Banane? (tuer, mitte, fenster) '),
    read(Ort2),nl,
    write('Wo befindet sich die Kiste? (tuer, mitte, fenster) '),read(Ort3),nl,
    (not(Ort1==Ort3),Kiste='auf_boden';
    write('Steht der Affe auf der Kiste? (auf_boden, auf_kiste) '),
    read(Kiste)),nl.
```

Regelbasierte Systeme: Umsetzung in Prolog

```
kann_essen(zustand(affe(_,_,ja),banane(_),kiste(_)),[]).

kann_essen(Z1,[Aktion|A]):-
    zustandsaenderung(Z1,Aktion,Z2),kann_essen(Z2,A).

zustandsaenderung(zustand(affe(O,auf_kiste,nein),banane(O),kiste(O)),
    ergreift,
    zustand(affe(O,auf_kiste,ja),banane(O),kiste(O))).

zustandsaenderung(zustand(affe(O,auf_boden,B),banane(O),kiste(O)),
    klettert,
    zustand(affe(O,auf_kiste,B),banane(O),kiste(O))).
```

Regelbasierte Systeme: Umsetzung in Prolog

```
zustandsaenderung(zustand(affe(O1,auf_boden,B),banane(O2),kiste(O1)),
    schiebt(O1,O2),
    zustand(affe(O2,auf_boden,B),banane(O2),kiste(O2))):-not(O1 == O2).

zustandsaenderung(zustand(affe(O1,auf_boden,B),banane(O),kiste(O2)),
    geht(O1,O2),
    zustand(affe(O2,auf_boden,B),banane(O),kiste(O2))):-not(O1 == O2).

zustandsaenderung(zustand(affe(O1,auf_kiste,B),banane(O),kiste(O1)),
    steigt_herab,
    zustand(affe(O1,auf_boden,B),banane(O),kiste(O1))).
```

Regelbasierte Systeme: Umsetzung in Prolog

```
schreibe_pfad([]):-nl.
schreibe_pfad([A|Rest]):-
    schreibe_aktion(A),nl,schreibe_pfad(Rest).
schreibe_aktion(ergreift):-write('Der Affe ergreift die Banane').
schreibe_aktion(klettert):-write('Der Affe klettert auf die Kiste').
schreibe_aktion(steigt_herab):-write('Der Affe steigt von der Kiste herunter').
schreibe_aktion(schiebt(A,B)):-
    write('Der Affe schiebt die Kiste'),
    von(A,A1),
    write(A1),
    zu(B, B1),
    write(B1).

schreibe_aktion(geht(A,B)):-
    write('Der Affe geht'),
    von(A,A1),
    write(A1),
    zu(B,B1),
    write(B1).
```

Regelbasierte Systeme: Umsetzung in Prolog

```
von(fenster, ' vom Fenster ').
von(tuer, ' von der Tür ').
von(mitte, ' von der Mitte ').
zu(fenster, 'zum Fenster').
zu(tuer, 'zur Tür').
zu(mitte, 'zur Mitte').
```

Prädikatenlogik: Parsing als Deduktion

Satz $\rightarrow$ NP$\cap$VP

NP$\rightarrow$ Det $\cap$ N

NP$\rightarrow$ Name

VP $\rightarrow$ Vt $\cap$ NP

VP $\rightarrow$ Vi

Det $\rightarrow$ the

N $\rightarrow$ *boy, girl, ball*

Name $\rightarrow$ *John, Mary*

Vt $\rightarrow$ *loves, kicked*

Vi $\rightarrow$ *jumped, cried*

Prädikatenlogik: Parsing als Deduktion

R1: $\bigwedge x \bigwedge y\ (NP(x) \wedge VP(x) \Rightarrow Satz(x \cap y))$

R2: $\bigwedge x \bigwedge y\ (Det(x) \wedge N(y) \Rightarrow NP(x \cap y))$

R3: $\bigwedge x\ (Name(x) \Rightarrow NP(x))$

R4: $\bigwedge x \bigwedge y\ (Vt(x) \wedge NP(x) \Rightarrow VP(x \cap y))$

R5: $\bigwedge x\ (Vi(x) \Rightarrow VP(x))$

Prädikatenlogik: Parsing als Deduktion

Lexikon:

- Det(the)
- N(boy)
- N(girl)
- N(ball)
- Name(John)
- Name(Mary)
- Vt(loves)
- Vt(kicked)
- Vi(jumped)
- Vi(cried)

Logisches Schließen in der Prädikatenlogik

Konjunktion

Sind P und Q Axiome, dann kann die Konjunktion $P \wedge Q$ zur Axiomenmenge hinzugefügt werden

Allbeseitigung

Da eine allquantifizierte Aussage für alle Individuen eines Individuenbereiches gelten soll, muss sie auch für ein einzelnes Individuum gelten.

Ist $\bigwedge x\, p(x)$ ein Axiom, dann kann die Aussage $p(a)$ zur Axiomenmenge hinzugefügt werden, wenn a zum Individuenbereich von x gehört.

Logisches Schließen in der Prädikatenlogik

Modus Ponens

Modus Ponens ist eines der bekanntesten Schluss-Schemata. Es hat die folgende Form:

$p \Rightarrow q$

p

$\therefore q$

Ein gültiges Schluss-Schema geht bei Ersetzung der Aussagenvariablen in einen gültigen Schluss über.

Prädikatenlogik: Parsing als Deduktion

Theorem: *Satz(the*∩girl∩cried)

Beweis:

(1) Det(the)	Lexikon
(2) N(girl)	Lexikon
(3) Det(the) ∧ N(girl)	(1), (2) Konjunktion
(4) Det(the) ∧ N(girl) ⇒ NP(the∩girl)	R2, Allbeseitigung
(5) NP(the∩girl)	(3), (4)Modus Ponens
(6) Vi(cried)	Lexikon
(7) Vi(cried) ⇒ VP(cried)	R5, Allbeseitigung
(8) VP(cried)	(6), (7)Modus Ponens
(9) NP(the∩girl) ∧ VP(cried)	(5), (8) Konjunktion
(10) NP(the∩gitl) ∧ VP(cried) ⇒Satz(the∩girl∩cried)	R1
(11) Satz(the∩girl∩cried)	(9), (10) Modus Ponens

Prädikatenlogik: Parsing als Deduktion

Definition 1. *Literal*

Ein Literal ist eine Primformel oder die Negation einer Primformel

Beispiele: NP(x), $\neg$ VP(y)

Prädikatenlogik: Parsing als Deduktion

Definition 2. *Klausel*

Eine Klausel ist eine Formel der Form $\bigwedge x_1 \ldots \bigwedge x_s\ (L_1 \vee \ldots \vee L_m)$, wobei jedes L_i ein Literal ist und $x_1 \ldots x_s$ die einzigen Variablen sind, die in $L_1 \vee \ldots \vee L_m$ vorkommen.

Klauselnotation:

$\bigwedge x_1 \ldots \bigwedge x_s (A_1 \vee \ldots \vee A_k \vee \neg B_1 \vee \ldots \vee \neg B_n)$:

$A_1, \ldots, A_k \Leftarrow B_1 \wedge \ldots \wedge B_n$

Prädikatenlogik: Parsing als Deduktion

Definition 3. *Programmklausel*

Eine Programmklausel ist eine Klausel der Form
$A \Leftarrow B_1, \ldots, B_n$

Definition 4. *Einheitsklausel*

Eine Einheitsklausel ist eine Klausel der Form
$A \Leftarrow$
d.h. eine Programmklausel ohne Rumpf.

Definition 5. *Zielklausel*

Eine Zielklausel ist eine Klausel der Form
$\Leftarrow B_1, \ldots, B_n$
d.h. eine Klausel ohne Kopf.

Prädikatenlogik: Parsing als Deduktion

Definition 6. *Horn Klausel*

Eine Horn Klausel (= *definite clause*) ist eine Klausel, die entweder eine Programmklausel oder eine Zielklausel ist.

Definition 7. *Logikprogramm*

Ein Logikprogramm ist eine endliche Menge von Programmklauseln.

Definition 8. *Definition*

In einem Logikprogramm ist die Menge aller Programmklauseln mit dem gleichen Prädikat p im Kopf die Definition von p.

Prädikatenlogik: Parsing als Deduktion

	Formel	**Kommentar**
0	$\bigwedge x \bigwedge y(NP(x) \wedge VP(y) \Rightarrow Satz(x \cap y))$	Ausgangsformel
1	$\bigwedge x \bigwedge y(\neg(NP(x) \wedge VP(y)) \vee Satz(x \cap y))$	Konditional
2	$\bigwedge x \bigwedge y(\neg(NP(x) \vee VP(y) \vee Satz(x \cap y))$	Skopus der Negation
3	$\neg NP(x) \vee \neg VP(y) \vee Satz(x \cap y)$	Präfix weglassen
4	$Satz(x \cap y) \vee \neg NP(x) \vee \neg VP(y)$	Ordnung der Literale
5	$Satz(x \cap y) \Leftarrow NP(x), VP(y)$	Klauselnotation

Die Gesamtgrammatik in konjunktiver Normalform

Sie lautet wie folgt, wobei die Variablen für spätere Referenzzwecke durch Indizes umbenannt werden:

R1: $\neg NP(x_1) \vee \neg VP(y_1) \vee Satz(x_1 \cap y_1)$

R2: $\neg Det(x_2) \vee \neg N(y_2) \vee NP(x_2 \cap y_2)$

R3: $\neg Name(x_3) \vee NP(x_3)$

R4: $\neg Vt(x_4) \vee \neg NP(y_4) \vee VP(x_4 \cap y_4)$

R5: $\neg Vi(x_5) \vee VP(x_5)$

Lexikon:

Det(the)	Name(John)	
N(boy)	Name(Mary)	
N(girl)	Vt(loves)	Vi(jumped)
N(ball)	Vt(kicked)	Vi(laughed)

PS-Grammatik in Klauselnotation

Bei der Umwandlung in Klauselnotation ist nur zu beachten, dass Lexikoneinträge positive Literale sind und daher zu Einheitsklauseln werden:

R1: $Satz(x_1 \cap y_1) \Leftarrow NP(x_1), VP(y_1)$

R2: $NP(x_2 \cap y_2) \Leftarrow Det(x_2), N(y_2)$

R3: $NP(x_3) \Leftarrow Name(x_3)$

R4: $VP(x_4 \cap y_4) \Leftarrow Vt(x_4), NP(y_4)$

R5: $VP(x_5) \Leftarrow Vi(x_5)$

PS-Grammatik in Klauselnotation

Lexikon:

Det(the) ⇐
N(boy) ⇐
N(girl) ⇐
N(ball) ⇐
Name(John) ⇐
Name(Mary) ⇐
Vt(loves) ⇐
Vt(kicked) ⇐
Vi(jumped) ⇐
Vi(laughed) ⇐

PS-Grammatik in Klauselnotation

Aus dieser Form der Grammatik ist zweierlei zu erkennen:

1. Alle Klauseln sind Programmklauseln oder Einheitsklauseln, d.h. die Grammatik ist ein Logikprogramm im definierten Sinne.
2. PS-Regeln im üblichen Format haben eigentlich im Kern bereits die Form von Programmklauseln. In einer PS-Regel wie $A \rightarrow B$ entspricht A einem positiven Literal und B einer Folge von negativen Literalen als Rumpf der Klausel.

Resolutionsschema

Damit das Resolutionsschema angewandt werden kann, ist erforderlich, dass in zwei verschiedenen Klauseln ein Literal einmal positiv und einmal negativ vorkommt.

$$\frac{p \vee q \qquad \neg p \vee r}{\therefore q \vee r}$$

Hier zeigt sich der syntaktische Vorteil von Programm-Klauseln, insofern nur der Kopf ein positives Literal sein kann, während der Rumpf nur aus negativen Literalen besteht. Zur Beseitigung eines Literals aus dem Rumpf einer Klausel müssen wir versuchen, dieses mit dem Kopf einer Programmklausel zu unifzieren.

$$\frac{p \Leftarrow q, r \qquad s \Leftarrow p, t}{s \Leftarrow q, r, t}$$

Substitution und Unifikation

Für die Anwendung des Resolutionsprinzips auf zwei Klauseln ist Voraussetzung, dass ein Literal in einer Klausel positiv, in der anderen negativ vorkommt. Im Rahmen der Prädikatenlogik entsteht ein Problem dadurch, dass Formeln erst durch die Substitution von Variablen vergleichbar werden.

Beispiel:

$$\neg \mathrm{Vi}(x_5) \vee \mathrm{VP}(x_5)$$

$$\mathrm{Vi}(\textit{laughed})$$

Das Resolutionsschema kann hier erst angewandt werden, wenn man die Variable x_5 durch *laughed* substituiert.

$$\neg \mathrm{Vi}(\textit{laughed}) \vee \mathrm{VP}(\textit{laughed})$$

$$\underline{\mathrm{Vi}(\mathrm{laughed})}$$

$$\mathrm{VP}(\textit{laughed}) \qquad \text{Resolvente}$$

Substitution und Unifikation

Das Verfahren, durch das festgestellt wird, ob zwei Ausdrücke durch geeignete Substitutionen für ihre Variablen gleich gemacht werden können, nennt man Unifikation. Die Möglichkeit der Unifikation ist eine Grundvoraussetzung für die Anwendung des Resolutionsprinzips in der Prädikatenlogik.

PS-Grammatik in Klauselnotation

R1: $Satz(x_1 \cap y_1) \Leftarrow NP(x_1), VP(y_1)$

R2: $NP(x_2 \cap y_2) \Leftarrow Det(x_2), N(y_2)$

R3: $NP(x_3) \Leftarrow Name(x_3)$

R4: $VP(x_4 \cap y_4) \Leftarrow Vt(x_4), NP(y_4)$

R5: $VP(x_5) \Leftarrow Vi(x_5)$

Lexikon:

$Det(the) \Leftarrow$

$N(boy) \Leftarrow$

$N(girl) \Leftarrow$

$N(ball) \Leftarrow$

$Name(John) \Leftarrow$

$Name(Mary) \Leftarrow$

$Vt(loves) \Leftarrow$

$Vt(kicked) \Leftarrow$

$Vi(jumped) \Leftarrow$

$Vi(laughed) \Leftarrow$

PS-Grammatik in Klauselnotation

Die Prämissen sind die Programmklauseln (einschließlich Einheitsklauseln) der Grammatik. Gemäß dem Verfahren des indirekten Beweises nehmen wir zunächst die Negation der zu beweisenden Aussage zu den Prämissen hinzu:

$\neg Satz(the \cap girl \cap laughed)$

Es handelt sich um ein negatives Literal, so dass wir die Klauselnotation

$\Leftarrow Satz(the \cap girl \cap laughed)$

d.h. eine Zielklausel erhalten.

Resolutionsschema in Aktion

Z: $\Leftarrow$ Satz(John.kicked.the.ball.nil,nil)

P: Satz(x_1,z_1) $\Leftarrow$ NP(x_1,y_1), VP(y_1,z_1)

U: {x_1/John.kicked.the.ball.nil, z_1/nil}

R: NP(John.kicked.the.ball.nil,y_1), VP(y_1,nil)

Z: =R

P: NP(x_3,z_3) $\Leftarrow$ Name(x_3,z_3)

U: {z_3/John.kicked.the.ball.nil, z_3/y_1}

R: $\Leftarrow$ Name(John.kicked.the.ball.nil, y_1}, VP(y_1, nil)

Z: =R

P: Name(John.z_{10}, z_{10}) $\Leftarrow$

U: {z_{10}/kicked.the.ball.nil, y_1/kicked.the.ball.nil}

R: $\Leftarrow$ VP(kicked.the.ball.nil, nil)

Z: =R

P: VP(x_4, z_4) $\Leftarrow$ Vt(x_4, y_4), NP(y_4, z_4)

U: {x_4/kicked.the.ball.nil,z_4/nil}

R: $\Leftarrow$ Vt(kicked.the.ball.nil,y_4),NP(y_4,nil)

Resolutionsschema in Aktion

R:		$\Leftarrow$ Vt(kicked.the.ball.nil,y_4),NP(y_4,nil)
Z:	=R	
P:	Vt(kicked.z_{13},z_{13})	$\Leftarrow$
U:	{z_{13}/the.ball.nil, y_4/the.ball.nil}	
R:		$\Leftarrow$ NP(the.ball.nil, nil)
Z:	=R	
P:	NP(x_2,z_2)	$\Leftarrow$ Det(x_2,y_2), N(y_2,z_2)
U:	{x_2/the.ball.nil, z_2/nil}	
R:		$\Leftarrow$ Det(the.ball.nil, y_2), N(y_2, nil)
Z:	=R	
P:	Det(the.z_6,z_6)	$\Leftarrow$
U:	{z_6/ball.nil, y_2/ball.nil}	
R:		$\Leftarrow$ N(ball.nil, nil)
Z:	=R	
P:	N(ball.z_9, z_9)	$\Leftarrow$
U:	{z_9/nil}	
R:		❑