DTAI

Declaratieve Talen en Artificiële Intelligentie

KU LEUVEN

The effects of buying a new car: An extension of the IDP KBS System

ICLP Technical Communication

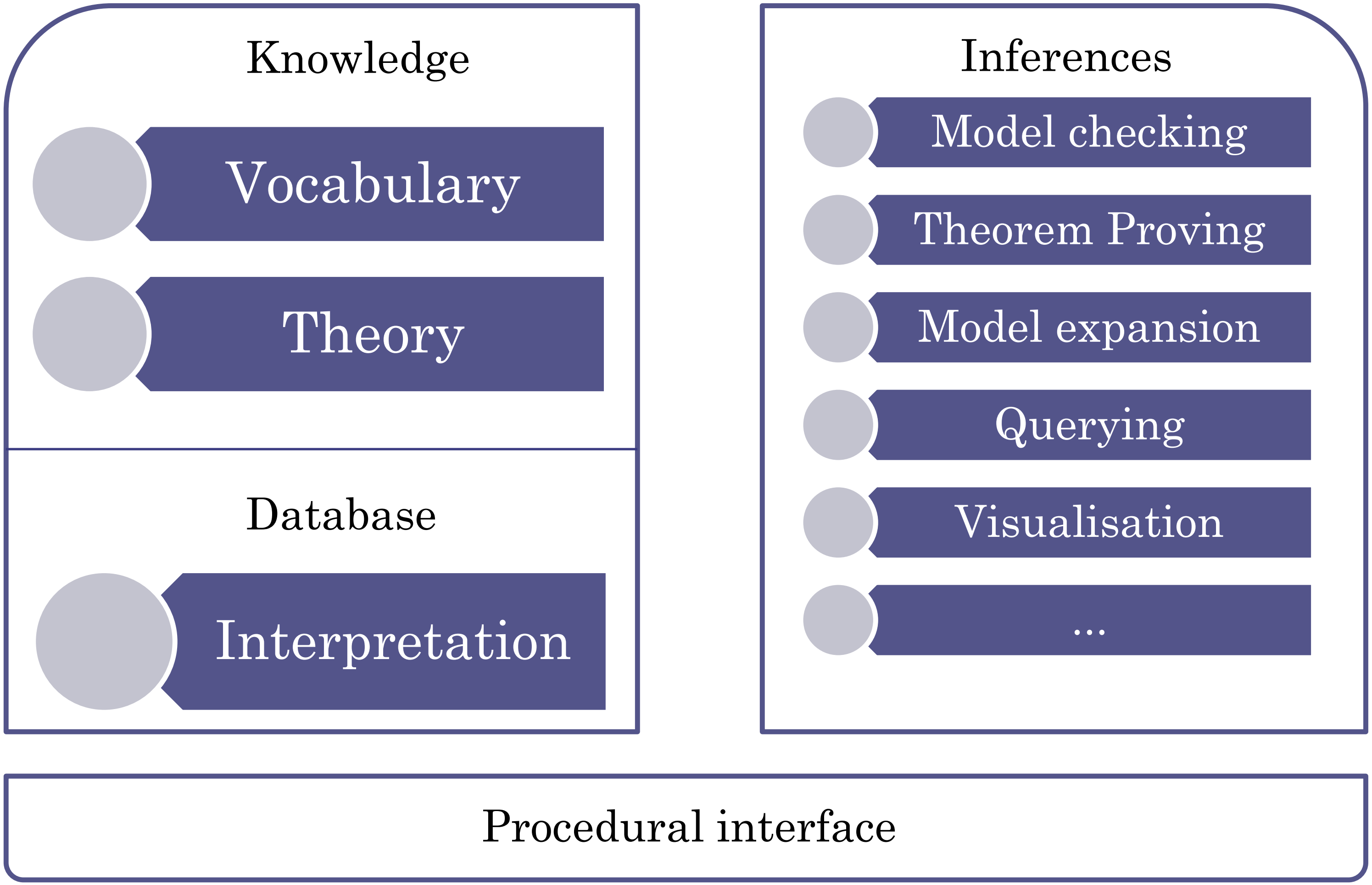
Pieter Van Hertum, Joost Vennekens, Bart Bogaerts, Jo Devriendt, Marc Denecker

Introduction: The IDP-system

- FO(.)
  - Extended FO
  - Types
  - Inductive definitions
  - Aggregates

- IDP2: Modelgenerator/expander
- IDP3: Knowledge Base System
  - Completely separate Knowledge from the problems you want to solve with it, and the inferences you want to run on it
  - Reuse for multiple applications!

Knowledge Base Paradigm



Goal of this talk

- Moment for evaluation of the system & the KBS-paradigm
- Take industry standard for knowledge-intense problems: **Business Rules**
- Take prototypical example of this standard: **EU-Rent Car Rental**

How does this application behave in our environment?

Business Rules

- Knowledge captured in procedural rules
- IF ………. THEN ……….
- Working memory contains collection of derived facts, which are used to choose new rule to fire.
- Rules are applied in forward reasoning till saturation

EU-Rent Car Rental

- Car Rental Company, consisting of different branches, each owning a collection of cars
- Tasks:
  - Planning Cars & handling reservations
  - Scheduling maintenance of Cars
  - Smaller database changes
    - Buying a new Car
    - Modifying details of an existing reservation
    - Deleting car because it is broken
    - …
  - …
- http://www.businessrulesgroup.org/first_paper/br01ad.htm

EU-Rent Car Rental

- Car Rental Company, consisting of different branches, each owning a collection of cars
- Tasks:
  - Planning Cars & handling reservations
  - Scheduling maintenance of Cars
  - Smaller database changes
    - Buying a new Car
    - Modifying details of an existing reservation
    - Deleting car because it is broken
    - ...
  - …
- http://www.businessrulesgroup.org/first_paper/br01ad.htm

Usecase 1: Planning of Cars in BRS

- Is the driver qualified?

IF ($ReservationClient(r, c)$ & $Age(c) > 18$)

THEN ($Insert(HasGoodDriver(r))$)

- Is there a car available?

IF ($ReservationCar(r, c)$ & $\neg(Overlap(r, r1)$ & $ReservationCar(r1, c)$ & $Accepted(r1))$

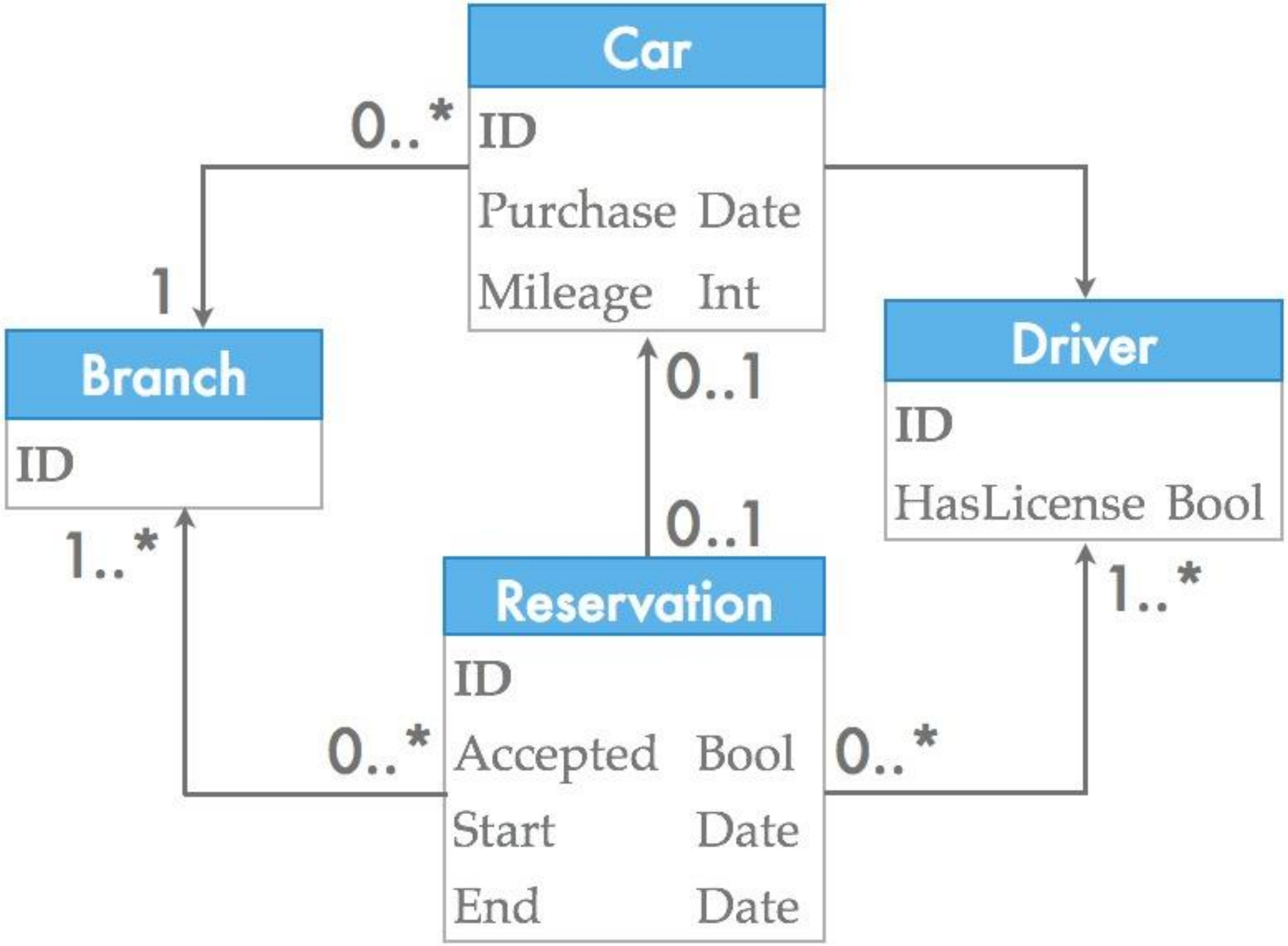
THEN ($Insert(HasGoodCar(r,c))$)

- What does overlap mean?

IF ($Start(r2) \leq End(r1)$ & $Start(r1) \leq End(r2)$)

THEN ($Insert(Overlap(r1,r2))$)

Representing a state of the system



Representing objects: Vocabularium

Vocabularium V{

type Car

type Driver

…

hasLicense(Driver)

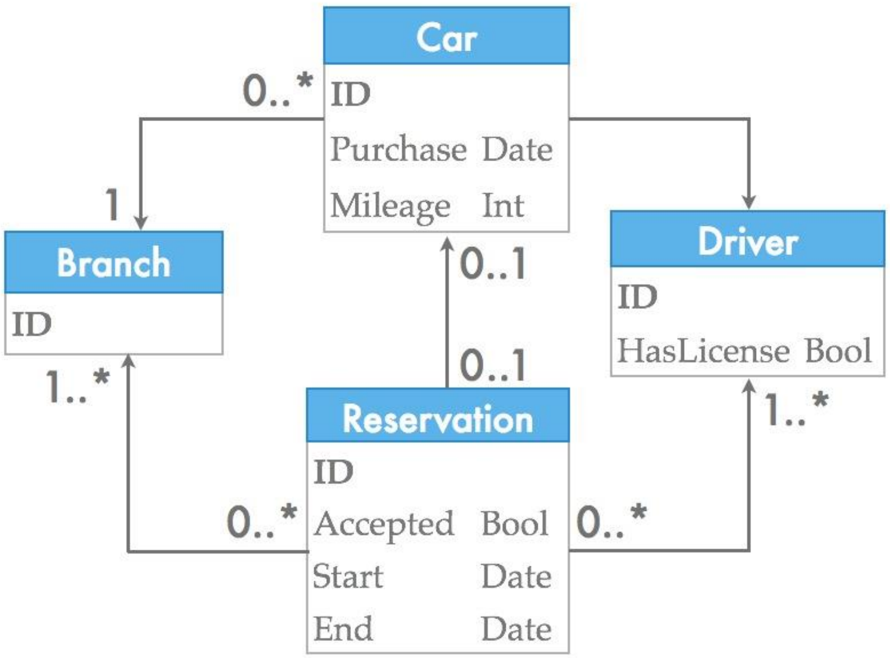
Accepted(Reservation)

…

Mileage: Car →Int

Location: Car→Branch

}



CONSTRAINTS TO BE A VALID STATE

- Every accepted reservation has to have a licensed driver.

  $\forall r : Accepted(r) \Rightarrow HasLicense(ResDriver(r)).$

- If a car is allocated to a reservation, it has to be at the right branch when the reservation starts.

  $\forall r\ c : Allocated(r,c) \Rightarrow$
  $Location(c,StartDate(r)) = Location(r).$

- There can only be 1 car allocated to each reservation.

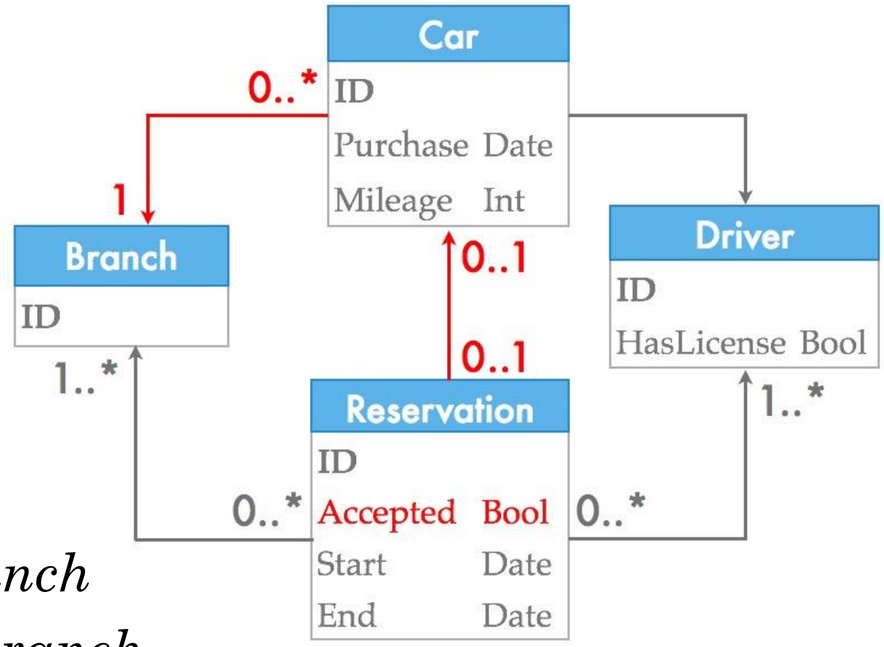
  $\forall r : \#\{c : Allocated(r,c)\} \leq 1$

Generating a plan

- Use Valid State theory
- Modelexpansion gives a valid planning
  - System is free to decide which reservations should be accepted
  - Possible choice: Decline all reservations
- Modelexpansion is not enough!
- New inference: **Optimizing Modelexpansion**
  - Generate plan with as much accepted reservations as possible

Dynamic

- Static theory not enough in changing environment
- Make Valid State time-dependant, because every state has to be a valid state.
  - *Accepted(Reserv, State)*
  - *Location: (Car, State)→Branch*
  - *Allocate: (Reserv, State)→Branch*



$$\left\{ \begin{array}{ll} \forall r\ t : Accepted(r, t+1) & \leftarrow Accepted(r,t) \wedge \neg Cancelled(r,t) \\ \forall t : Accepted(NewRes(t), t) & \leftarrow \exists c : Allocated(NewRes(t), c, t). \end{array} \right\}$$

Are we there yet?

- Optimization inference + dynamic model
- System can decide to cancel all reservations between states and accepted an all new set of reservations
- We need to penalize changing parts of the model when this is not necessary
- New inference: **Weighted Model Revision**

Weighted Model Revision

- Add a weight to making a symbol more true/false
- Maximize weight
- *Accepted→{100,-300} : Accepting a reservation will make a profit of $100, however cancelling one costs $300*
- *Location → (−50,−50): Moving a car costs $50*
- *Allocated → (−20,−20): Changing a car in a reservation costs $20*

Usecase 2: Buying the car

- Trivial problem in BRS
- Closed world assumption → Domain fixed
- If we buy a new car, we have to “break it open”
- We extended our language: complex heads in definitions

$$\left\{\begin{array}{c} \forall s\ sn : \textbf{new}\ c : \mathrm{SerialNr}(c) = sn \land \\ \mathrm{PurchaseDate}(c) = s \land \mathrm{Mileage}(c) = 0 \\ \leftarrow \mathrm{BuyCar}(sn,s). \end{array}\right\}$$

- Syntax & Semantics: FO(ID+)

Conclusions

- Implementing BRS using the KBS-paradigm : a test using IDP3
- We implemented 2 use cases
  - Use case 1 Reservation use case:
    - IDP3 language simpler specification than in BRS
    - IDP3 model expansion/optimisation suitable form of inference

      → **IDP3 offers better solution than BRS**
  - Use case 2: buying a new car
    - IDP3 not suitable for introducing new objects

      → **Extension of language is required**
- Several usecases remain: *To Be Continued*…