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

Knowledge
- Vocabulary
- Theory
- Database
- Interpretation

Inferences
- Model checking
- Theorem Proving
- Model expansion
- Querying
- Visualisation
- ...

Procedural interface
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 \((\text{ReservationClient}(r, c) \& \text{Age}(c) > 18)\)
  THEN \((\text{Insert}(\text{HasGoodDriver}(r)))\)

- Is there a car available?
  IF \((\text{ReservationCar}(r, c) \& \neg(\text{Overlap}(r, r1) \&
    \text{ReservationCar}(r1, c) \& \text{Accepted}(r1)))\)
  THEN \((\text{Insert}(\text{HasGoodCar}(r,c)))\)

- What does overlap mean?
  IF \((\text{Start}(r2) \leq \text{End}(r1) \& \text{Start}(r1) \leq \text{End}(r2))\)
  THEN \((\text{Insert}(\text{Overlap}(r1,r2)))\)
REPRESENTING A STATE OF THE SYSTEM

- **Branch**
  - ID
  - 1..* Reservation
  - 1..* Car

- **Car**
  - ID
  - Purchase Date
  - Mileage
  - Int
  - 0..* Reservation
  - 0..* Driver

- **Reservation**
  - ID
  - Accepted
  - Start
  - End
  - 0..* Driver
  - 0..* Car

- **Driver**
  - ID
  - HasLicense
  - Bool
  - 0..* Car
  - 0..* Reservation
  - 1..* Driver
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\((\text{Reserv, State})\)
  - Location: \((\text{Car, State})\rightarrow\text{Branch}\)
  - Allocate: \((\text{Reserv, State})\rightarrow\text{Branch}\)

\[
\begin{align*}
\forall r \ t : \text{Accepted}(r, t + 1) & \iff \text{Accepted}(r, t) \land \neg \text{Cancelled}(r, t) \\
\forall t : \text{Accepted}(\text{NewRes}(t), t) & \iff \exists c : \text{Allocated}(\text{NewRes}(t), c, t).
\end{align*}
\]
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

\[
\forall s \text{ sn : new c : SerialNr}(c) = sn \land \\
\text{PurchaseDate}(c) = s \land \text{Mileage}(c) = 0 \\
\leftarrow\text{BuyCar}(sn,s).
\]

- 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 use cases remain: *To Be Continued...*