SUPPLEMENTARY MATERIAL

COMPLETE DIALOGUE

A dialogue excerpt is shown here as an example of the type of dialogues that can take place in systems developed with the approach put forward in this paper. We do so to demonstrate that we can support all the desired features of intelligent dialogue systems discussed in Section 4.3. In fact, it goes further allowing also ontological discussions. This case study includes the following agents:

assistant (a): the internal representation in MAS for a chatbot that assists hospital staff in carrying out bed allocation in a hospital;
operator (o): the internal representation in MAS for the hospital staff member who operates the system for allocating bed;
nurse (n): the internal representation in MAS for a nurse who in that hospital serves as domain expert for bed allocation and whom the operator needs to consult in case of doubt;
database (d): an agent that has access to the hospital general information system for checking details of past and current patients, bed allocations, etc.
ontology (on): an agent expert in ontologies, responsible for semantic reasoning using argumentation schemes as defeasible rules generated automatically from the semantic rules contained in the ontology.
optimiser (op): an agent responsible for making suggestions for optimised allocations using the GLPSol solver of GLPK.

We show each (numbered) dialogue game move but *before* it we provide an English equivalent for readability. We enclose in curly brackets the changing beliefs of some of the agents, which underlie the dialogue move. Note that our approach only allows for atomic formulæ in argument conclusions, but it allows for constraints on a particular conclusion to be specified using Jason annotations, so if a dialogue move contains a formula `p(X)[q(X)]`, it means that in Jason we will find an instantiation for `X` such that `p(X) & q(X)` holds.

The dialogue starts with the operator trying to allocate a bed to a particular patient, proceeds as follows.

operator to assistant: check if any female surgical bed is free;
`1. question(o,a,free(B)[female(B), surgical(B)])`
{assistant: `des(o,allocate(P,B)[female(B),surgical(B)]), bel(d,free(B))`}
assistant to database: check if any female surgical bed is free;
`2. question(a,d,free(B)[female(B),surgical(B)])`
database to assistant: bed 203b is free and satisfies those constraints;
`3. assert(d,a,free(203b)[female(203b),surgical(203b)])`
assistant to operator: bed 203b has the characteristics you requested;
```
4. assert(a,o,answer(free(203b)[female(203b),
surgical(203b)])
```
operator to nurse: I'm allocating Patient8 to 203b;
`5. assert(o,n,allocate(patient8,203b))`
{assistant: `des(o,allocate(patient8,B)[female(B),surgical(B)])`}
nurse to operator: we cannot allocate patient8 to 203b;
`6. assert(n,o,~allocate(patient8,203b))`
operator to nurse: why?
`7. challenge(o,n,~allocate(patient8,203b))`
nurse to all: this bed is in a room that has many beds, for Patient8 we need the smallest room with the fewest occupied beds;
```
8. justify(n,*,[defeasible_rule(
~allocate(patient8,203b),[large(203),in_room(203b,
203)])[as(nurse_statement)],
defeasible_rule(allocate(patient8,B),
[in_room(B,R),smallest(R)[fewest_occupants(R)]])
[as(nurse_restriction)]])
```
{assistant: `des(n,allocate(patient8,B)[female(B),surgical(B),`
`in_room(B,R),smallest(R)[fewest_occupants(R)]]),`
`defeasible_rule(allocate(patient8,B),[in_room(B,R), smallest(R)[fewest_occupants(R)]])[as(nurse_restriction)]`
operator to assistant: how about allocating Patient8 to the bed that was freed yesterday by Patient6;
```
9. assert(o,a,allocate(patient8,B)
[allocated(patient6,B,TI,TF),
within_time(yesterday,TI,TF)])
```
{assistant: `des(n,allocate(patient8,B)[female(B),surgical(B),`
`in_room(B,R),smallest(R)[fewest_occupants(R)]]),`
`defeasible_rule(allocate(patient8,B),[in_room(B,R), smallest(R)[fewest_occupants(R)]])[as(nurse_restriction)],`
`des(o,allocate(patient8,B)[allocated(patient6,B,TI,TF), within_time(yesterday,TI,TF)]),`
`bel(d,allocated(P,B,TI,TF))`}

assistant to database: which bed did Patient6 free yesterday?
```
10. question(a,d,allocated(patient6,B,TI,TF)
[within_time(yesterday,TI,TF)])
```
database to assistant: Patient6 was allocated to bed 202b;
```
11. assert(d,a,allocated(patient6,202b,t1,t2))
```
assistant to ontology: Is bed 202b suitable to Patient8?
```
12. question(a,on,question(suitable(202b,patient8)))
```
ontology to assistant: No, it is not.
```
13. assert(on,a,~suitable(202b,patient8))
```
assistant to operator: the bed freed by patient Patient6 is 202b; it is not suitable for patient Patient8;
```
14. assert(a,o,answer(allocated(patient6,202b,t1,t2)
15. assert(a,o,answer(~suitable(202b,patient8))
```
operator to assistant: why not?
```
16. challenge(o,a,~suitable(202b,patient8))
```
entering an ontological subdialogue using OAsdlg2
```
17. ontoargsubdlg(a,*,~suitable(202b,patient8))
```
assistant to ontology: Explain why bed 202b is not suitable to Patient8.
```
17a. question(a,on,explain(~suitable(202b,patient8)))
```
ontology to assistant: Patient Patient8 is of the age group Adult and bed 202b is of the age group Adolescent that is different from Adult. So bed 202b is unsuitable for patient Patient8.
```
17b. assert(on,a,defeasible_rule(
~suitable(202b,patient8),[patient(patient8),
hospital_bed(202b),is_of_the_age_group(patient8,adult),
bed_is_of_the_age_group(202b,adolescent),
differentFrom(adult,adolescent)])[as(nSbyAG)])
```
assistant to all: Patient Patient8 is of the age group Adult and bed 202b is of the age group Adolescent that is different from Adult. So bed 202b is unsuitable for patient Patient8. `17c. assert(a,*,answer(defeasible_rule(`
```
~suitable(202b,patient8),[patient(patient8),
hospital_bed(202b),is_of_the_age_group(patient8,
adult),bed_is_of_the_age_group(202b,adolescent),
differentFrom(adult,adolescent)])[as(nSbyAG)])
```
operator to all: Why do you think bed 202b is of the age group Adolescent?
```
17d. challenge(o,*,
bed_is_of_the_age_group(202b,adolescent))
```
assistant to ontology: Explain why bed 202b is of the age group Adolescent.
```
17e. question(a,on,explain(bed_is_of_the_age_group(
202b,adolescent)))
```
ontology to assistant: Patient Patient5 is of the age group Adolescent and is in the same room as bed 202b
```
17f. assert(on,a,defeasible_rule(
bed_is_of_the_age_group(202b,adolescent),
[patient(patient5),is_of_the_age_group(patient5,
adolescent),hospital_bed(202a),
occupy_one(patient5,202a),is_in(202a,202),
hospital_bed(202b),is_in(202b,202)])[as(aGbyPinRoom)])
```
assistant to all: Patient Patient5 is of the age group Adolescent and is in the same room as bed 202b
```
17g. justify(a,*,answer(defeasible_rule(
bed_is_of_the_age_group(202b,adolescent),
[patient(patient5),is_of_the_age_group(patient5,
adolescent),hospital_bed(202a),
occupy_one(patient5,202a),is_in(202a,202),
hospital_bed(202b),is_in(202b,202)])[as(aGbyPinRoom)])
```
nurse to all: we can make an exception in this case, they can stay in the same room provided they are of the same gender and same type of care;
```
17h. assert(n,*,defeasible_rule(suitable(B,patient8),
[patient(patient8),bed(B),is_of_the_gender(patient8,G),
bed_is_of_gender(B,G),is_of_care(patient8,C),
bed_is_of_care(B,C)])[as(nurse_exception)])
```

{assistant: `des(n,defeasible_rule(suitable(B,patient8), [patient(patient8),bed(B),is_of_the_gender(patient8,G), bed_is_of_gender(B,G),is_of_care(patient8,C), bed_is_of_care(B,C)])[as(nurse_exception)])`}

assistant to ontology: Nurse made an exception; Is bed 202b suitable to Patient8 now?

```
17i. assert(a,on,defeasible_rule(suitable(B,patient8),
[patient(patient8),bed(B),is_of_the_gender(patient8,G),
bed_is_of_gender(B,G),is_of_care(patient8,C),
bed_is_of_care(B,C)])[as(nurse_exception)])
22a. question(a,on,question(suitable(202b,patient8)))
```

ontology to assistant: No, it is not.

```
17j. assert(on,a,~suitable(202b,patient8))
```

assistant to all: that room is still not suitable;

```
17k. assert(a,*,~suitable(202b,patient8))
```

operator to all: why not?

```
17l. challenge(o,*,~suitable(202b,patient8))
```

assistant to ontology: Explain why bed 202b is still not suitable to patient Patient8

```
17m. question(a,on,explain(~suitable(202b,patient8)))
```

ontology to assistant: Patient Patient8 is of Intensive care and bed 202b is of Minimal care that is different from Intensive care. So bed 202b is unsuitable for patient Patient8

```
17n. assert(on,a,defeasible_rule(
is_unsuitable_for(202b,patient8),[patient(patient8),
hospital_bed(202b),is_care(patient8,intensive),
bed_is_care(202b,minimal),
differentFrom(intensive,minimal)])[as(nSbyCare)]
```

assistant to all: Patient Patient8 is of Intensive care and bed 202b is of Minimal care that is different from Intensive care. so bed 202b is unsuitable for patient Patient8

```
17o. justify(a,*,defeasible_rule(
is_unsuitable_for(202b,patient8),[patient(patient8),
hospital_bed(202b),is_care(patient8,intensive),
bed_is_care(202b,minimal),
differentFrom(intensive,minimal)])[as(nSbyCare)]
```

... **NB:** We suppressed messages from all agents accepting; `~suitable(202b,patient8)`.

closing ontological subdialogue in agreement that `~suitable(202b,patient8)` using rule CloseOAsdlg1 as all agree with that content, the one used to open the subdialogue;

```
17p. closesubdlg(a,*,~suitable(202b,patient8))
```

{assistant: `des(o,allocate(patient8,B)[female(B),surgical(B)])`}

assistant to operator: Would you like me to try to suggest another bed for Patient8?

```
18. question(a,o,des(o,suggestion(B,patient8)
[suitable(B,patient8)]))
```

operator to assistant: yes, please!

```
19. assert(o,a,des(o,suggestion(B,patient8)
[suitable(B,patient8)]))
```

{assistant: `des(n,defeasible_rule(allocate(patient8,B), [in_room(B,R),smallest(R)[fewest_occupants(R)]]]) [as(nurse_restriction)]), des(o,suggestion(B,patient8)[suitable(B,patient8)])`

assistant to optimiser: generate an allocation suggestion to patient Patient8 considering the nurse restriction.

```
20. question(a,op,suggestion(B,patient8)[suitable(B,
patient8),defeasible_rule(C,R)[1][as(nurse_restriction)]]])
```

optimiser to assistant: I can not generate a suggestion to patient Patient8 considering this restriction.

```
21. assert(op,a,~suggestion(B,patient8)
[defeasible_rule(C,R]])[as(nurse_restriction)]]]])
```

{assistant:`des(n,defeasible_rule(suitable(B,patient8), [patient(patient8),bed(B),is_of_the_gender(patient8,G), bed_is_of_gender(B,G),is_of_care(patient8,C), bed_is_of_care(B,C)])[as(nurse_exception)])`}

assistant to all: can I use the exception made by nurse?

```
22. question(a,*,des(o,suggestion(B,patient8)
[suitable(B,patient8),
```

[1]We use the abbreviation C to represent the rule's conclusion and R to represent the body of the rule described just before.

defeasible_rule(Ce,Re)[as(nurse_exception)]]))
nurse to all: yes, you can.
23. assert(n,*,des(o,suggestion(B,patient8)
[suitable(B,patient8),
defeasible_rule(Ce,Re)[as(nurse_exception)]]))
assistant to optimiser: Generate a suggestion considering the nurse restriction and the nurse exception.
24. question(a,op,suggestion(B,patient8)[suitable(B,
patient8),defeasible_rule(Ce,Re)[as(nurse_exception)],
defeasible_rule(Cr,Rr)[as(nurse_restriction)]])
optimiser to assistant: Considering this restrictions and exceptions bed 201a is suitable for Patient8
25. assert(op,a,suggestion(201a,patient8)
[suitable(201a,patient8)])
{assistant: suitable(202a,patient8)}
assistant to all: Considering the exception made by nurse I suggest allocating Patient8 to bed 201a
26. assert(a,*,suggestion(201a,patient8)
[suitable(201a,patient8),
[defeasible_rule(Ce,Re)[as(nurse_exception)]])
operator to assistant: ok, please book bed 201a for Patient8 who will leave the operation room not before 19:00 nor after 20:30;
27. assert(o,a,booked(201a, patient8, 19:00h, 20:30h))
{assistant: des(o,booked(201a, patient8, 19:00h, 20:30h))}
assistant to database: book bed 201a to patient Patient8 from 19:00H and keep the booking until 20:30H;
28. assert(a,d, booked(201a, patient8, 19:00h, 20:30h))
database to assistant: Ok, bed 201a is now booked;
29. accept(d,a,booked(201a,patient8,19:00h,20:30h))
{assistant: booked(201a, patient8, 19:00h, 20:30h)}
assistant to operator: booking done!
30. assert(a,o,booked(201a,patient8,19:00h,20:30h))
operator to assistant: Ok!
31. accept(o,a,booked(201a,patient8,19:00h,20:30h))