SUPPLEMENTARY MATERIAL LARA – Human-guided collaborative problem solver: Effective integration of learning, reasoning and communication

Appendix A. Dialogue Manager

Dialogue manager in LARA maintains a fixed set of template sentences with slots shown below. As required the slots are populated and the natural language sentence generated is sent to the architect.

Unknown shape I do not understand this shape. Try building it using tower or row.
Was the shape misspelled? May be try building a row
System does not support this shape. Do you wanna build a row instead?

missing spatial relation
(Can you describe|Could you tell me|Can you tell me) where the {new_structure} is placed with respect to the {existing_structure} (we just built)?

missing dimension
(What is the|What's the|Can you describe the|Could you tell me the) {missing_dimension} of the {new_structure}?

Parse error for spatial relation
Could not understand the spatial relationship. Can you say something like: The left-end of the row is on the east of the top-end of tower
Embarassed, I do not understand that.

Parse error
Sorry, I had trouble understanding that. Could you explain it differently?
Sorry, I don't understand. Can you try again?
Sorry, I'm having trouble understanding. Could you reword that?

Planning error
Sorry, I'm not able to do that. Could we try again?
Sorry, I'm not able to build that. Could you reword that?
Sorry, I can't do that. Could you explain it differently?

Parse Clarification
Sorry, I had trouble understanding that.
Sorry, I don't understand.
Sorry, I'm having trouble understanding.

Initial Greeting
Hi Architect, what are we building today?
I'm ready! What are we building?
Hello! What are we building?
Hello Architect, I'm ready!

Next Greeting
Hi Architect, what are we building today?
I'm ready! What are we building?
Hello! What are we building?
Hello Architect, I'm ready!

Next Prompts
Okay, what's next?
Okay, now what?
What are we doing next?

Appendix B. Background file

The complete list of predicates used in the FOL language can be found in the background file furnished below.

```
setParam: nodeSize=100.
setParam: loadAllBasicModes = false.



// Parts

// Shapes
mode: row(+Part).
mode: column(+Part).
mode: tower(+Part).
mode: square(+Part).
mode: rectangle(+Part).
mode: cube(+Part).
mode: cuboid(+Part).
```

```
mode: block(+Block).
mode: blockS(+Part).

// Dimensions
mode: width(+Part, #FloatPart).
mode: height(+Part, #FloatPart).
mode: length(+Part, #FloatPart).
mode: size(+Part,#FloatPart).

// Properties
mode: color(+Part, #ColorPart).
mode: spatial_rel(&rel,+Loc,+Loc).
mode: location(+Part).


// relation

mode: top_behind_left(+Part,-Block).
mode: top_left_behind(+Part,-Block).
mode: behind_top_left(+Part,-Block).
mode: behind_left_top(+Part,-Block).
mode: left_behind_top(+Part,-Block).
mode: left_top_behind(+Part,-Block).
mode: top_behind_right(+Part,-Block).
mode: top_right_behind(+Part,-Block).
mode: behind_top_right(+Block,-Block).
mode: behind_right_top(+Part,-Block).
mode: right_behind_top(+Part,-Block).
mode: right_top_behind(+Part,-Block).
mode: top_front_left(+Part,-Block).
mode: top_left_front(+Part,-Block).
mode: front_top_left(+Part,-Block).
mode: front_left_top(+Part,-Block).
mode: left_front_top(+Part,-Block).
mode: left_top_front(+Part,-Block).
mode: top_front_right(+Part,-Block).
mode: top_right_front(+Part,-Block).
mode: front_top_right(+Part,-Block).
mode: front_right_top(+Part,-Block).
mode: right_front_top(+Part,-Block).
mode: right_top_front(+Part,-Block).
mode: bottom_behind_left(+Part,-Block).
```

```
mode: bottom_left_behind(+Part,-Block).
mode: behind_bottom_left(+Part,-Block).
mode: behind_left_bottom(+Part,-Block).
mode: left_behind_bottom(+Part,-Block).
mode: left_bottom_behind(+Part,-Block).
mode: bottom_behind_right(+Part,-Block).
mode: bottom_right_behind(+Part,-Block).
mode: behind_bottom_right(+Part,-Block).
mode: behind_right_bottom(+Part,-Block).
mode: right_behind_bottom(+Part,-Block).
mode: right_bottom_behind(+Part,-Block).
mode: bottom_front_left(+Part,-Block).
mode: bottom_left_front(+Part,-Block).
mode: front_bottom_left(+Part,-Block).
mode: front_left_bottom(+Part,-Block).
mode: left_front_bottom(+Part,-Block).
mode: left_bottom_front(+Part,-Block).
mode: bottom_front_right(+Part,-Block).
mode: bottom_right_front(+Part,-Block).
mode: front_bottom_right(+Part,-Block).
mode: front_right_bottom(+Part,-Block).
mode: right_front_bottom(+Part,-Block).
mode: right_bottom_front(+Part,-Block).
mode: behind_left(+Part,-Block).
mode: left_behind(+Part,-Block).
mode: behind_right(+Part,-Block).
mode: right_behind(+Part,-Block).
mode: front_left(+Part,-Block).
mode: left_front(+Part,-Block).
mode: front_right(+Part,-Block).
mode: right_front(+Part,-Block).
mode: left_end(+Part,-Block).
mode: right_end(+Part,-Block).
mode: front_end(+Part,-Block).
mode: behind_end(+Part,-Block).
mode: top_end(+Part,-Block).
mode: bottom_end(+Part,-Block).
mode: block_location(+Block,-Loc).

// Bridgers

bridger: contains/2.
```

```
bridger: spatial_rel/3.

// Precomputes
mode: sameColor(+ColorShape,+ColorPart).
mode: sameSP(+FloatShape,+FloatPart).
mode: sameSS(+FloatShape,+FloatPart).
//mode: samePP(+FloatPart,+FloatPart).
mode: oneMoreSP(+FloatShape,+FloatPart).
mode: oneMorePS(+FloatPart,+FloatShape).
//mode: oneMorePP(+FloatPart,+FloatPart).
mode: oneMoreSS(+FloatShape,+FloatShape).

precompute: sameColor(X, Y) :- colorShape(Shape,X), color(Part,Y),
    ↪  X is Y.
precompute: sameSP(X, Y) :- heightShape(Shape,X), height(Part,Y),
    ↪ sameAs(X, Y).
precompute: sameSP(X, Y) :- widthShape(Shape,X), width(Part,Y),
    ↪ sameAs(X, Y).
precompute: sameSP(X, Y) :- lengthShape(Shape,X), length(Part,Y),
    ↪ sameAs(X, Y).
precompute: sameSP(X, Y) :- sizeShape(Shape,X), size(Part,Y),
    ↪ sameAs(X, Y).
precompute: sameSS(X, Y) :- heightShape(Shape,X), widthShape(Shape
    ↪ ,Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- heightShape(Shape,X), lengthShape(
    ↪ Shape,Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- heightShape(Shape,X), sizeShape(Shape,
    ↪ Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- widthShape(Shape,X), lengthShape(Shape
    ↪ ,Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- widthShape(Shape,X), sizeShape(Shape,Y
    ↪ ), sameAs(X, Y).
precompute: sameSS(X, Y) :- lengthShape(Shape,X), sizeShape(Shape,
    ↪ Y), sameAs(X, Y).

//precompute: samePP(X, Y) :- height(Part1,X), width(Part2,Y),
    ↪ sameAs(X, Y).
//precompute: samePP(X, Y) :- height(Part1,X), width(Part2,Y),
    ↪ sameAs(X, Y).
//precompute: oneMorePP(X, Y) :- height(Part1,X), width(Part2,Y),
    ↪ minus(X, Y, Z), Z is 1.
```

```
//precompute: oneMorePP(X, Y) :- height(Part1,X), height(Part2,Y),
   ↪  minus(X, Y, Z), Z is 1.
//precompute: oneMorePP(X, Y) :- width(Part1,X), height(Part2,Y),
   ↪ minus(X, Y, Z), Z is 1.

precompute: oneMoreSS(X, Y) :- heightShape(Shape,X), heightShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- heightShape(Shape,X), widthShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- heightShape(Shape,X), lengthShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- heightShape(Shape,X), sizeShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- widthShape(Shape,X), heightShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- widthShape(Shape,X), widthShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- widthShape(Shape,X), lengthShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- widthShape(Shape,X), sizeShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- lengthShape(Shape,X), heightShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- lengthShape(Shape,X), widthShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- lengthShape(Shape,X), lengthShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- lengthShape(Shape,X), sizeShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- sizeShape(Shape,X), heightShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- sizeShape(Shape,X), widthShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- sizeShape(Shape,X), lengthShape(
   ↪ Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- sizeShape(Shape,X), sizeShape(Shape
   ↪ ,Y), minus(X, Y, Z), Z is 1.


precompute: oneMoreSP(X, Y) :- heightShape(Shape,X), height(Part,Y
   ↪ ), minus(X, Y, Z), Z is 1.
```

```
precompute: oneMoreSP(X, Y) :- heightShape(Shape,X), width(Part,Y)
   ↪ , minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- heightShape(Shape,X), length(Part,Y
   ↪ ), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- heightShape(Shape,X), size(Part,Y),
   ↪  minus(X, Y, Z), Z is 1.

precompute: oneMoreSP(X, Y) :- widthShape(Shape,X), height(Part,Y)
   ↪ , minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- widthShape(Shape,X), width(Part,Y),
   ↪  minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- widthShape(Shape,X), length(Part,Y)
   ↪ , minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- widthShape(Shape,X), size(Part,Y),
   ↪ minus(X, Y, Z), Z is 1.

precompute: oneMoreSP(X, Y) :- lengthShape(Shape,X), height(Part,Y
   ↪ ), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- lengthShape(Shape,X), width(Part,Y)
   ↪ , minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- lengthShape(Shape,X), length(Part,Y
   ↪ ), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- lengthShape(Shape,X), size(Part,Y),
   ↪  minus(X, Y, Z), Z is 1.

precompute: oneMoreSP(X, Y) :- sizeShape(Shape,X), height(Part,Y),
   ↪  minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- sizeShape(Shape,X), width(Part,Y),
   ↪ minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- sizeShape(Shape,X), length(Part,Y),
   ↪  minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- sizeShape(Shape,X), size(Part,Y),
   ↪ minus(X, Y, Z), Z is 1.


precompute: oneMorePS(Y, X) :- heightShape(Shape,X), height(Part,Y
   ↪ ), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- heightShape(Shape,X), width(Part,Y)
   ↪ , minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- heightShape(Shape,X), length(Part,Y
   ↪ ), minus(Y, X, Z), Z is 1.
```

```
precompute: oneMorePS(Y, X) :- heightShape(Shape,X), size(Part,Y),
   ↪  minus(Y, X, Z), Z is 1.


precompute: oneMorePS(Y, X) :- widthShape(Shape,X), height(Part,Y)
   ↪ , minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- widthShape(Shape,X), width(Part,Y),
   ↪  minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- widthShape(Shape,X), length(Part,Y)
   ↪ , minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- widthShape(Shape,X), size(Part,Y),
   ↪ minus(Y, X, Z), Z is 1.

precompute: oneMorePS(Y, X) :- lengthShape(Shape,X), height(Part,Y
   ↪ ), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- lengthShape(Shape,X), width(Part,Y)
   ↪ , minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- lengthShape(Shape,X), length(Part,Y
   ↪ ), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- lengthShape(Shape,X), size(Part,Y),
   ↪  minus(Y, X, Z), Z is 1.

precompute: oneMorePS(Y, X) :- sizeShape(Shape,X), height(Part,Y),
   ↪  minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- sizeShape(Shape,X), width(Part,Y),
   ↪ minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- sizeShape(Shape,X), length(Part,Y),
   ↪  minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- sizeShape(Shape,X), size(Part,Y),
   ↪ minus(Y, X, Z), Z is 1.
```

Appendix C. Planner

Below we present the list of predicates in the JSHOP2 planner.

```
(row ?x-loc ?y-loc ?z-loc ?width ?color)
(tower ?x-loc ?y-loc ?z-loc ?height ?color)
(column ?x-loc ?y-loc ?z-loc ?length ?color)
(square ?x-loc ?y-loc ?z-loc ?width ?color)
(rectangle ?x-loc ?y-loc ?z-loc ?width ?height ?color)
(cube ?x-loc ?y-loc ?z-loc ?width ?color)
(cuboid ?x-loc ?y-loc ?z-loc ?height ?width ?length ?color)
```

```
(block ?x-loc ?y-loc ?z-loc ?color)
```