Causal Compiler Syntax

# RULES Block

All of your causal relations should be defined within a single RULES block.

Ex. 0

RULES {

/\* YOUR RELATIONS HERE \*/

}

# Causal Relations

A causal relation can be broken into an action implied by a set of other actions. Each action is defined by a set of parameters. A causal relation will be put on its own line and will appear similar to the following example:

Ex. 1

moveTo(obj, dest, dx, dy, dz, da) := grasp(obj), release(obj, dest, dx, dy, dz, da);

#### Actions

Each action in a causal relation has a set of parameters which allows the user to define certain characteristics of actions. In the following example, the action moveTo is defined as moving an object with name obj to a destination dest with destination coordinates and angle dx, dy, dz, and da.

Ex. 2

moveTo(obj, dest, dx, dy, dz, da)

#### Action Implications

Each causal relation defines an action as *implied* by a set of other actions. Implication is shown with the := operator.

Ex. 3

IMPLIED\_ACTION(PARAMETERS) := ACTION0(PARAMETERS),ACTION1(PARAMETERS);

# Conditional Causal Relations

Some causal relations may be defined as valid only under some pre-defined condition. This conditional will be defined with an if-block. Each if-block starts on its own line with the conditional statement. An indented causal relation will be placed on the next line. See example 4.

Ex. 4

if (CONDITIONAL\_STATEMENT):

CAUSAL\_RELATION

#### Conditionals

Conditionals are implemented similar to many languages and may be chained together as such using the && (logical AND) and || (logical OR) operators.

Ex. 5: The causal relation will only hold if CONDITION0 and CONDITION1 hold.

if (CONDITION0 && CONDITION1):

CAUSAL\_RELATION

#### Types of Conditionals

There are four types of comparisons that a user may use in a conditional statement. The variables used in these conditionals must be present in the parameters of the actions used in the associated CAUSAL\_RELATION.

1. **Variable Comparison**Compare two variables to each other. This is an identity comparison of the two variables. In the following comparison, if the object referenced by obj is the same object referenced by obj1 then the comparison is true.  
     
   Ex. 6  
    obj = obj1
2. **Literal Comparison**Similar to the variable comparison, the literal comparison compares a variable to a string literal. In the following comparison, if the string referenced by dest is equal to ‘room’ then the comparison is true.  
     
   Ex. 7  
    dest = ‘room’
3. **TYPE Operator**Get the TYPE of an object. This can then used as a comparison to some other value or another TYPE operator. The following example is a comparison checking if the TYPE of the object referenced by obj is block.  
     
   Ex. 8  
    TYPE(obj) = block
4. **ALL Operator**Get a list of all objects with a given TYPE. This can then be used as a comparison to a list of objects (See the section on Lists below). The following example is a comparison checking if obj1 and obj2 reference the only objects of TYPE block.  
     
   Ex. 9  
    ALL(block) = [obj1, obj2]

# Peripheral Syntax Features

1. **White Space**

The language is entirely whitespace agnostic.  
Ex. 10  
 ACTION0 (PARAMETERS)  
is equivalent to  
 ACTION0(PARAMETERS)

1. **Lists**Lists are implemented as comma-separated variables wrapped by brackets. They are used in conjunction with the ALL operator. See example 9 for typical usage.
2. **CONT# Operator**The CONT# operator can be used to continue a list of objects. The following example represents checking that the list of ALL objects of TYPE block does in fact contain obj1.  
     
   Ex. 11  
    ALL(block) = [obj1, CONT1]  
     
   A given CONT# operator must be fed the correct information via the # part of the operator. For the above example, the number 1 is passed as the list is a continuation from obj1. In the next example, we show a continuation from obj1 once more but this time within the set of parameters of an action stack. This can be interpreted as a “continuation” on obj1 but not on the rest of the parameters.  
     
   Ex. 12  
    stack(dest, dx, dy, dz, da, obj1, CONT1)

# Complete Example

Ex. 13

RULES {

move-to(obj, dest, dx, dy, dz, da) := grasp(obj), release(obj, dest, dx, dy, dz, da);

if (TYPE(obj)=block):

stack(dest, dx, dy, dz, da, obj) := move-to(obj, dest, dx, dy, dz, da);

if (TYPE(obj1) = block && obj = obj1):

stack(dest, dx, dy, dz, da, obj1, obj2, obj3, CONT3) := move-to(obj, dest, dx, dy, dz, da), stack(obj1, 0, 0, 0.5, 0, obj2, obj3, CONT2);

if (ALL(block)=[obj1, CONT1] && dest = 'room'):

stack-all(dx, dy, dz, da) := stack(dest, dx, dy, dz, da, obj1, CONT1)

}