Learning Rule Based Programming

Using Games

CashFlow Example

Classes

Account	
long	accountNo
int	balance

CashFlow	
Date	date
int	amount

AccountPeriod	
Date	start
Date	end

CashFlow Rule

```
select * from  Account acc,
     Cashflow cf, AccountPeriod ap
where acc.accountNo ==  cf.accountNo and
      cf.type == CREDIT
      cf.date >= ap.start and
      cf.date <= ap.end
acc.balance += cf.amount
```

```
rule “increase balance for AccountPeriod Credits”
    when
        ap  : AccountPeriod()
        acc : Account()
        cf  : CashFlow( type == CREDIT,
                        accountNo == acc.accountNo,
                        date >= ap.start && <= ap.end )
    then
        acc.balance  += cf.amount;
end
```

CashFlow Rule

```
select * from  Account acc,
     Cashflow cf, AccountPeriod ap
where acc.accountNo ==  cf.accountNo and
      cf.type == CREDIT
      cf.date >= ap.start and
      cf.date <= ap.end
acc.balance += cf.amount
```

```
rule “increase balance for AccountPeriod Credits”
    when
        ap  : AccountPeriod()
        acc : Account()
        cf  : CashFlow( type == CREDIT,
                        accountNo == acc.accountNo,
                        date >= ap.start && <= ap.end )
    then
        acc.balance  += cf.amount;
end
```

CashFlow Rule

```
select * from  Account acc,
     Cashflow cf, AccountPeriod ap
where acc.accountNo ==  cf.accountNo and
      cf.type == CREDIT
      cf.date >= ap.start and
      cf.date <= ap.end
acc.balance += cf.amount
```

```
rule “increase balance for AccountPeriod Credits”
    when
        ap  : AccountPeriod()
        acc : Account()
        cf  : CashFlow( type == CREDIT,
                        accountNo == acc.accountNo,
                        date >= ap.start && <= ap.end )
    then
        acc.balance  += cf.amount;
end
```

CashFlow Rule

```
select * from  Account acc,
     Cashflow cf, AccountPeriod ap
where acc.accountNo ==  cf.accountNo and
      cf.type == CREDIT
      cf.date >= ap.start and
      cf.date <= ap.end
acc.balance += cf.amount
```

```
rule “increase balance for AccountPeriod Credits”
    when
        ap  : AccountPeriod()
        acc : Account()
        cf  : CashFlow( type == CREDIT,
                        accountNo == acc.accountNo,
                        date >= ap.start && <= ap.end )
    then
        acc.balance  += cf.amount;
end
```

CashFlow Rule

```
select * from  Account acc,
     Cashflow cf, AccountPeriod ap
where acc.accountNo ==  cf.accountNo and
      cf.type == CREDIT
      cf.date >= ap.start and
      cf.date <= ap.end
acc.balance += cf.amount
```

```
rule “increase balance for AccountPeriod Credits”
    when
        ap  : AccountPeriod()
        acc : Account()
        cf  : CashFlow( type == CREDIT,
                        accountNo == acc.accountNo,
                        date >= ap.start && <= ap.end )
    then
        acc.balance  += cf.amount;
end
```

CashFlow Example

CashFlow			
date	amount	type	accountNo
12-Jan-12	100	CREDIT	1
2-Feb-12	200	DEBIT	1
18-May-12	50	CREDIT	1
9-Mar-12	75	CREDIT	1

AccountingPeriod	
start	end
01-JAN-2012	31-MAR-2012

Account	
accountNo	balance
1	0

```
rule "Increase balance for AccountPeriod Credits"
when
    ap : AccountPeriod( )
    acc : Account( )
    cf : CashFlow( type == CashFlowType.CREDIT,
                   accountNo == acc.accountNo,
                   date >= ap.start  && <= ap.end )
then
    acc.balance = acc.balance + cf.amount;
end
```

```
rule "Decrease balance for AccountPeriod Debits"
when
    ap : AccountPeriod( )
    acc : Account( )
    cf : CashFlow( type == CashFlowType.DEBIT,
                   accountNo == acc.accountNo,
                   date >= ap.start && <= ap.end )
then
    acc.balance = acc.balance - cf.amount;
end
```

CashFlow			
date	amount	type	accountNo
12-Jan-12	100	CREDIT	1
9-Mar-12	75	CREDIT	1

CashFlow			
date	amount	type	accountNo
2-Feb-12	200	DEBIT	1

Account	
accountNo	balance
1	-25

CashFlow Example

```
rule "Print blance for AccountPeriod" salience -50
when
        ap : AccountPeriod()
        acc : Account( )
then
        System.out.println( "Account Number " + acc.accountNo
+ " balance " + acc.balance );
end
```

Agenda		
1	increase balance	arbitrary
2	decrease balance	
3	increase balance	
4	print balance	

CashFlow Example

CashFlow			
date	amount	type	accountNo
12-Jan-12	100	CREDIT	1
2-Feb-12	200	DEBIT	1
18-May-12	50	CREDIT	1
9-Mar-12	75	CREDIT	1

AccountingPeriod	
start	end
01-Apr-2012	30-JUN-2012

Account	
accountNo	balance
1	0

```
rule "Increase balance for AccountPeriod Credits"
when
    ap : AccountPeriod( )
    acc : Account( )
    cf : CashFlow( type == CashFlowType.CREDIT,
                   accountNo == acc.accountNo,
                   date >= ap.start  && <=
ap.end )
then
    acc.balance = acc.balance + cf.amount;
end
```

```
rule "Decrease balance for AccountPeriod Debits"
when
    ap : AccountPeriod( )
    acc : Account( )
    cf : CashFlow( type == CashFlowType.DEBIT,
                   accountNo == acc.accountNo,
                   date >= ap.start && <= ap.end )
then
    acc.balance = acc.balance - cf.amount;
end
```

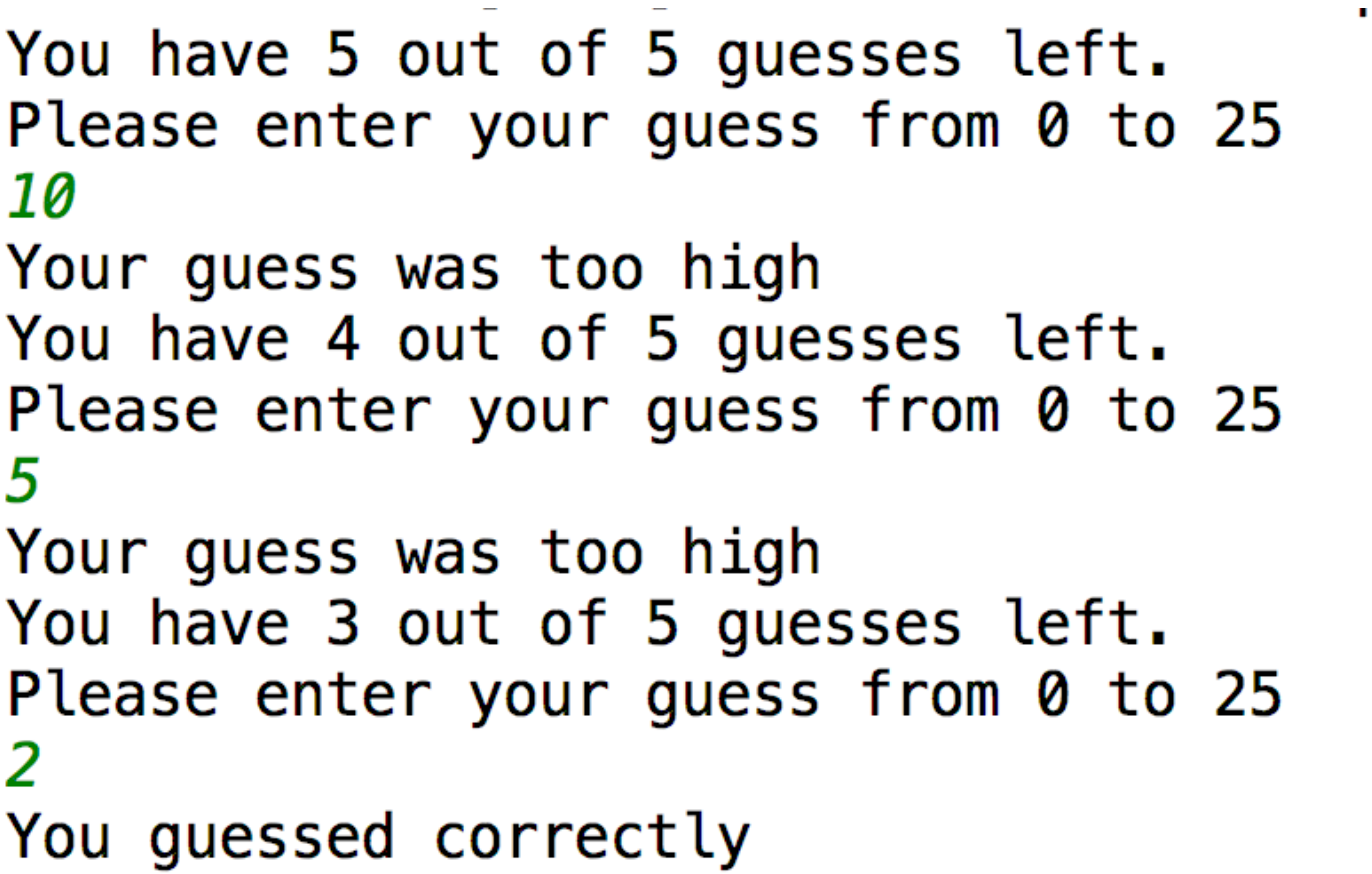
CashFlow			
date	amount	type	accountNo
18-May-12	75	CREDIT	1

CashFlow			
date	amount	type	accountNo

Account	
accountNo	balance
1	25

Number Guess

Number Guess



```
You have 5 out of 5 guesses left.
Please enter your guess from 0 to 25
10
Your guess was too high
You have 4 out of 5 guesses left.
Please enter your guess from 0 to 25
5
Your guess was too high
You have 3 out of 5 guesses left.
Please enter your guess from 0 to 25
2
You guessed correctly
```

```
 public class Game {
     private int biggest;
     private int smallest;
     private int guessCount;

 public class Guess {
     private int value;

public class GameRules {
    private int maxRange;
    private int allowedGuesses;


public class RandomNumber {
    private int randomNumber;
```

```java
public class Game {
    private int biggest;
    private int smallest;
    private int guessCount;
```

```java
public class GameRules {
    private int maxRange;
    private int allowedGuesses;
```

```java
public class RandomNumber {
    private int randomNumber;
```

```java
public class Guess {
    private int value;
```

```xml
<kbase name="NumberGuessKB" packages="org.drools.games.numberguess">
    <ksession name="NumberGuessKS"/>
</kbase>
```

```java
public class NumberGuessMain {

    public static void main(String[] args) {
        KieContainer kc = KieServices.Factory.get().getKieClasspathContainer();
        final KieSession ksession = kc.newKieSession( "NumberGuessKS");

        ksession.insert( new GameRules( 100, 5 ) );
        ksession.insert( new RandomNumber() );
        ksession.insert( new Game() );

        ksession.fireAllRules();
    }

}
```

```
rule Main when
    rules : GameRules( )
    game : Game( guessCount < rules.allowedGuesses )
    not Guess()
then
    setFocus("Guess");
end



rule "Get user Guess" agenda-group "Guess" when
    $r : RandomNumber()
    rules : GameRules( )
    game : Game( )
    not Guess()
then
    System.out.println( "You have " + ( rules.allowedGuesses - game.guessCount ) +
                        " out of " + rules.allowedGuesses +
                        " guesses left.\nPlease enter your guess from 0 to " +
                        rules.maxRange );

    br = new BufferedReader( new InputStreamReader( System.in ) );

    modify (game) { guessCount = game.guessCount + 1 }

    int i = Integer.parseInt( br.readLine() );
    insert( new Guess( i ) );
end
```

```
rule "Record the highest Guess" agenda-group "Guess"  no-loop when
    game  : Game( )
    r : RandomNumber()
    guess : Guess( value > r.value)
then
    modify ( game ) { biggest = guess.value };
    retract( guess );
    System.out.println( "Your guess was too high" );
end
```

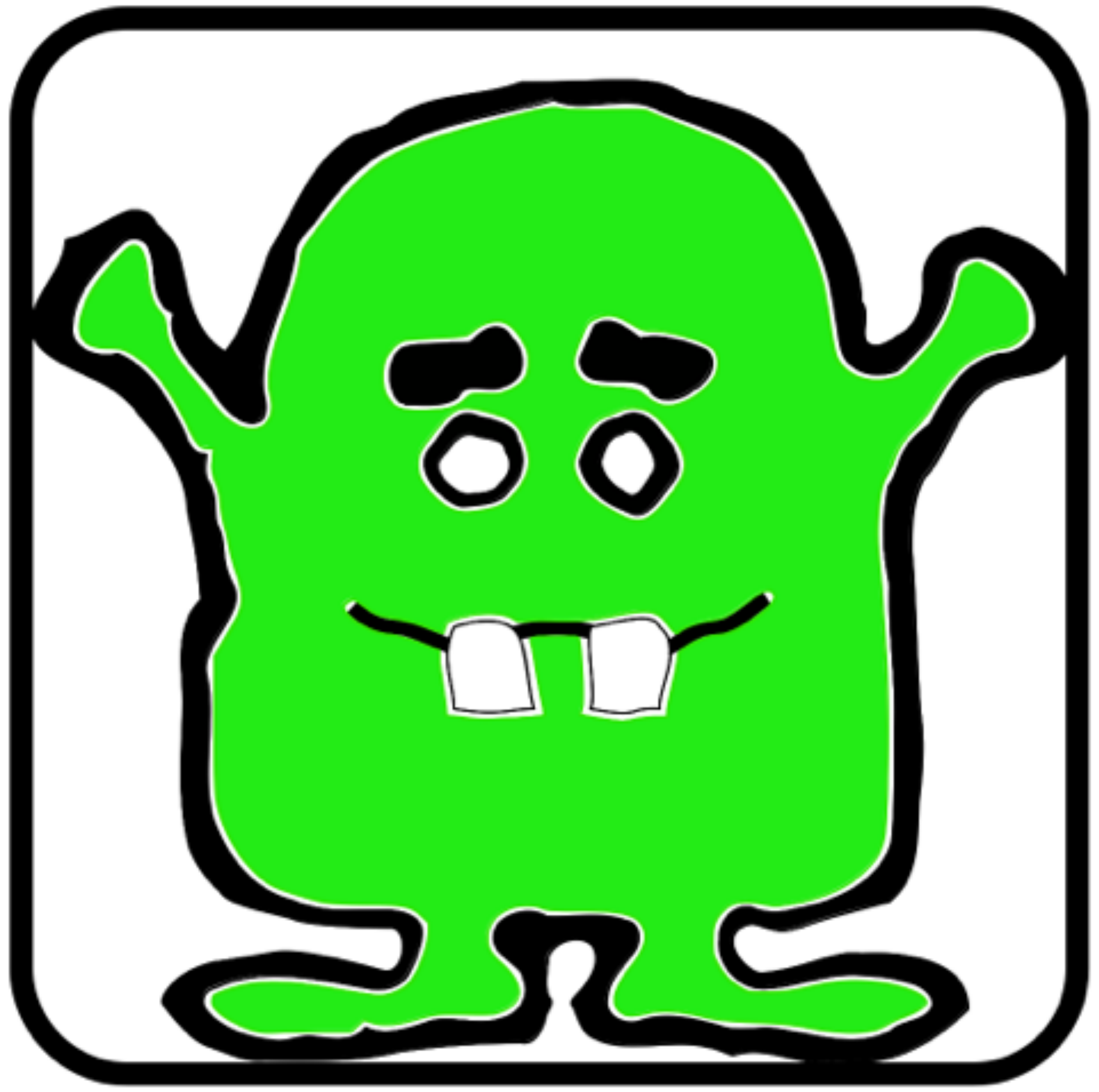
```
rule "Record the highest Guess" agenda-group "Guess"  no-loop when
    game  : Game( )
    r : RandomNumber()
    guess : Guess( value > r.value)
then
    modify ( game ) { biggest = guess.value };
    retract( guess );
    System.out.println( "Your guess was too high" );
end
```

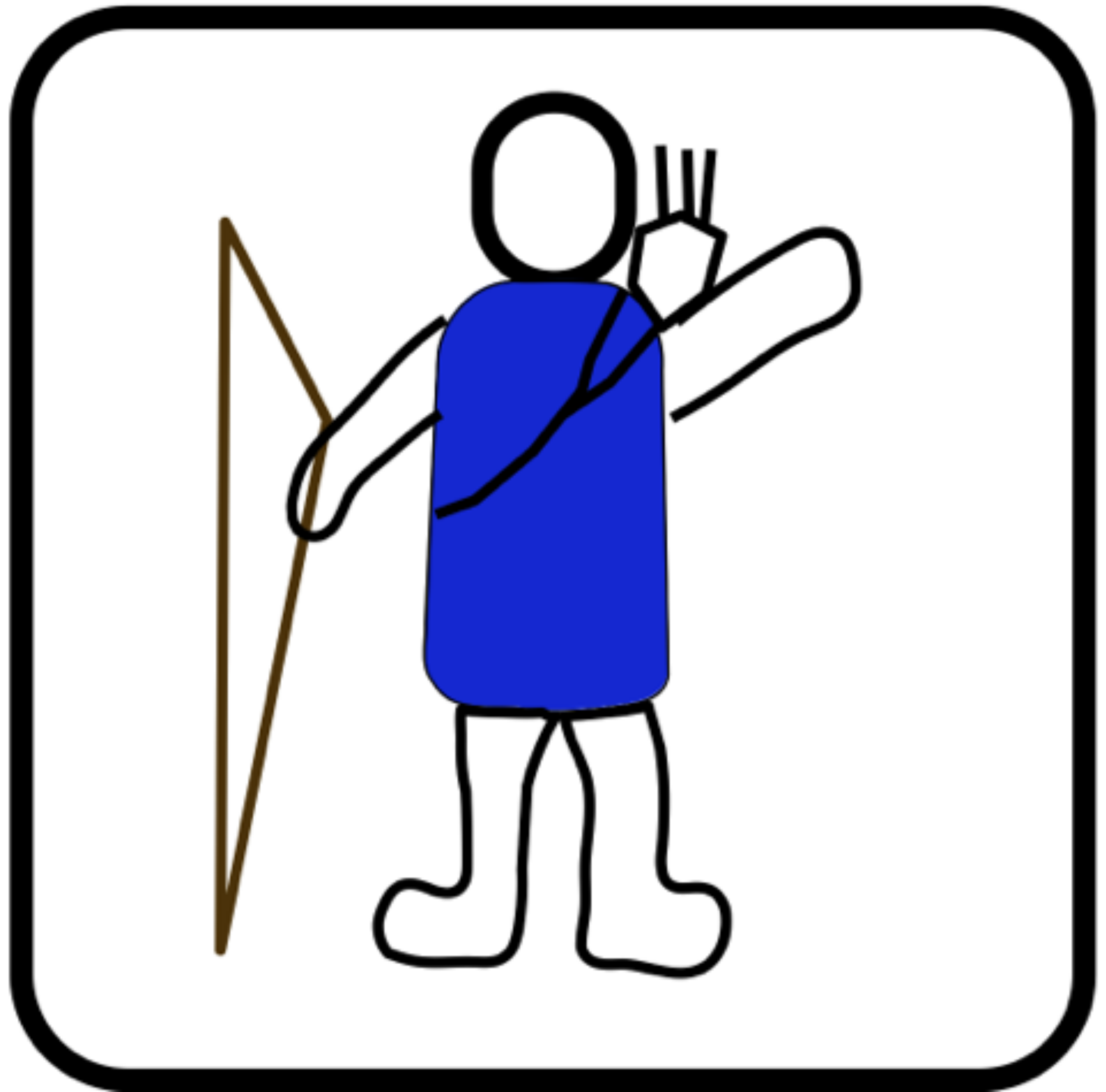
```
rule "Record the lowest Guess" agenda-group "Guess" when
    game : Game( )
    r : RandomNumber()
    guess :  Guess(value < r.value )
then
    modify ( game ) { smallest = guess.value };
    retract( guess );
    System.out.println( "Your guess was too low" );
end
```

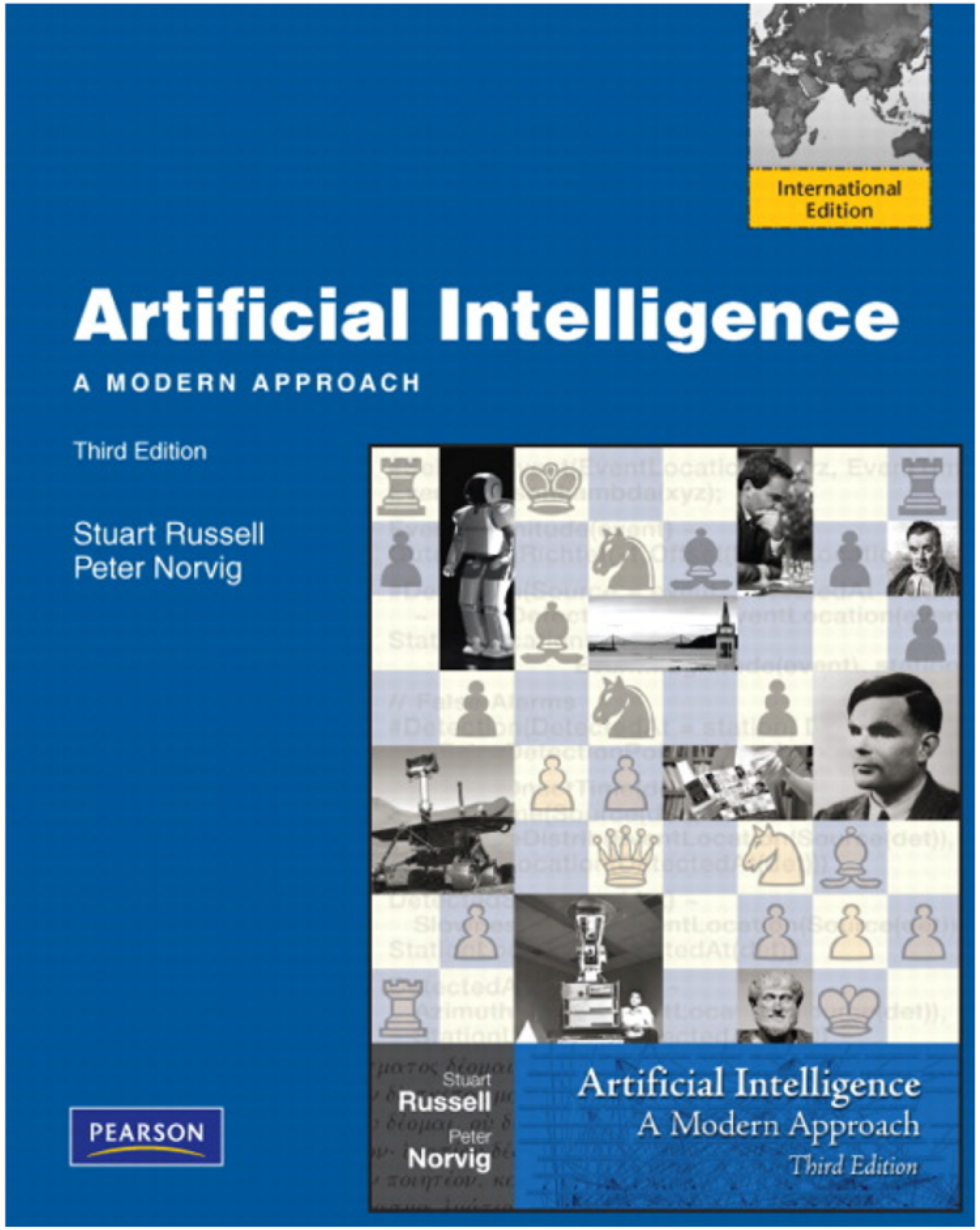
```
rule "Record the highest Guess" agenda-group "Guess"  no-loop when
    game  : Game( )
    r : RandomNumber()
    guess : Guess( value > r.value)
then
    modify ( game ) { biggest = guess.value };
    retract( guess );
    System.out.println( "Your guess was too high" );
end

rule "Record the lowest Guess" agenda-group "Guess" when
    game : Game( )
    r : RandomNumber()
    guess :  Guess(value < r.value )
then
    modify ( game ) { smallest = guess.value };
    retract( guess );
    System.out.println( "Your guess was too low"  );
end


rule "Guess correct" agenda-group "Guess" when
    game  : Game( )
    r : RandomNumber()
    guess : Guess( value == r.value)
then
    System.out.println( "You guessed correctly" );
end
```

```
rule Main when
    rules : GameRules( )
    game : Game( guessCount < rules.allowedGuesses )
    not Guess()
then
    setFocus("Guess");
end



rule "No more Guesses" when
    rules : GameRules( )
    game : Game( guessCount == rules.allowedGuesses )
    not Guess()
    r : RandomNumber()
then
    System.out.println( "You have no more guesses\nThe correct guess was " + r.value );
end
```

Wumpus







International Edition

Artificial Intelligence

A MODERN APPROACH

Third Edition

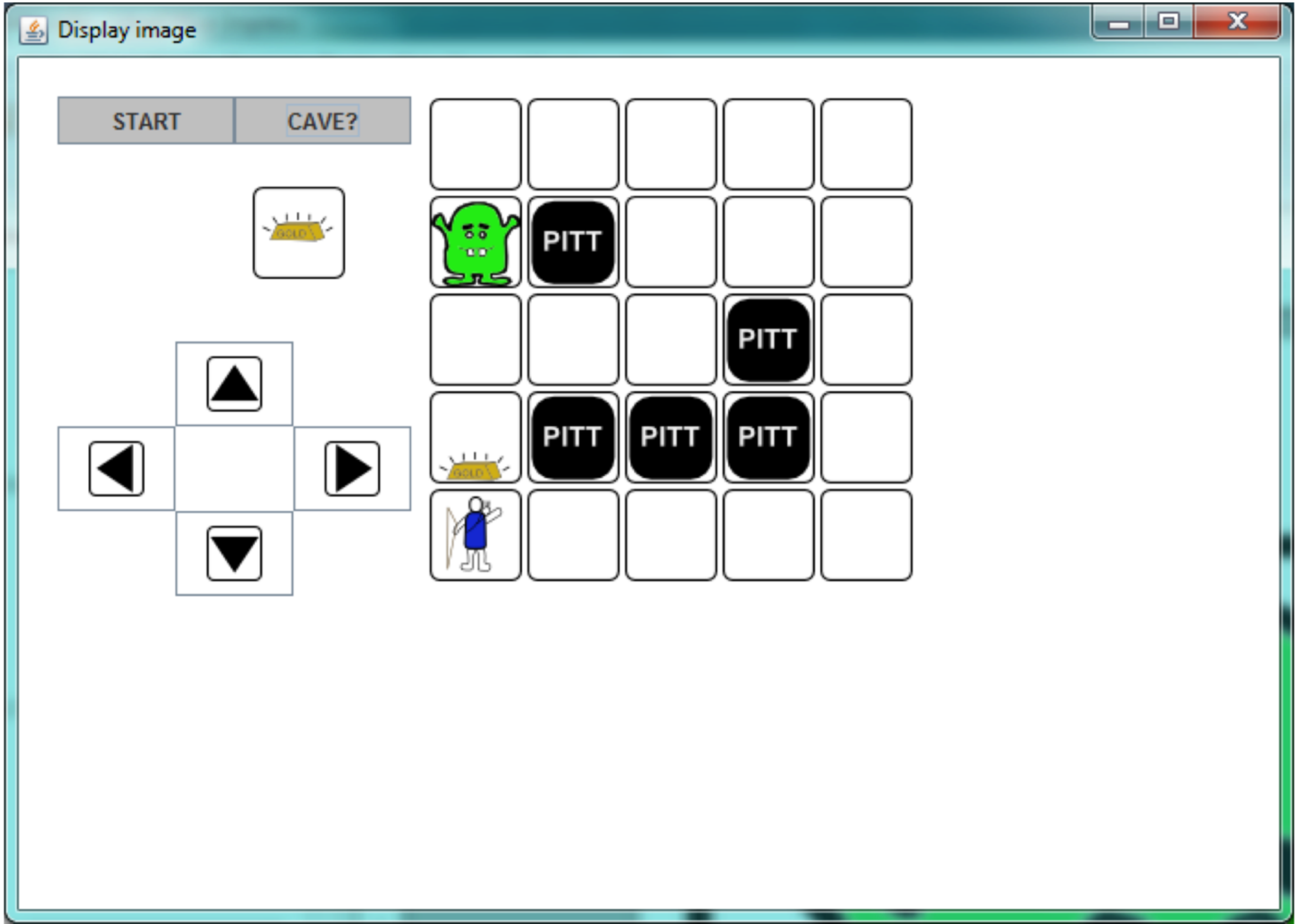
Stuart Russell
Peter Norvig

PEARSON

Stuart
Russell
Peter
Norvig

Artificial Intelligence
A Modern Approach
Third Edition

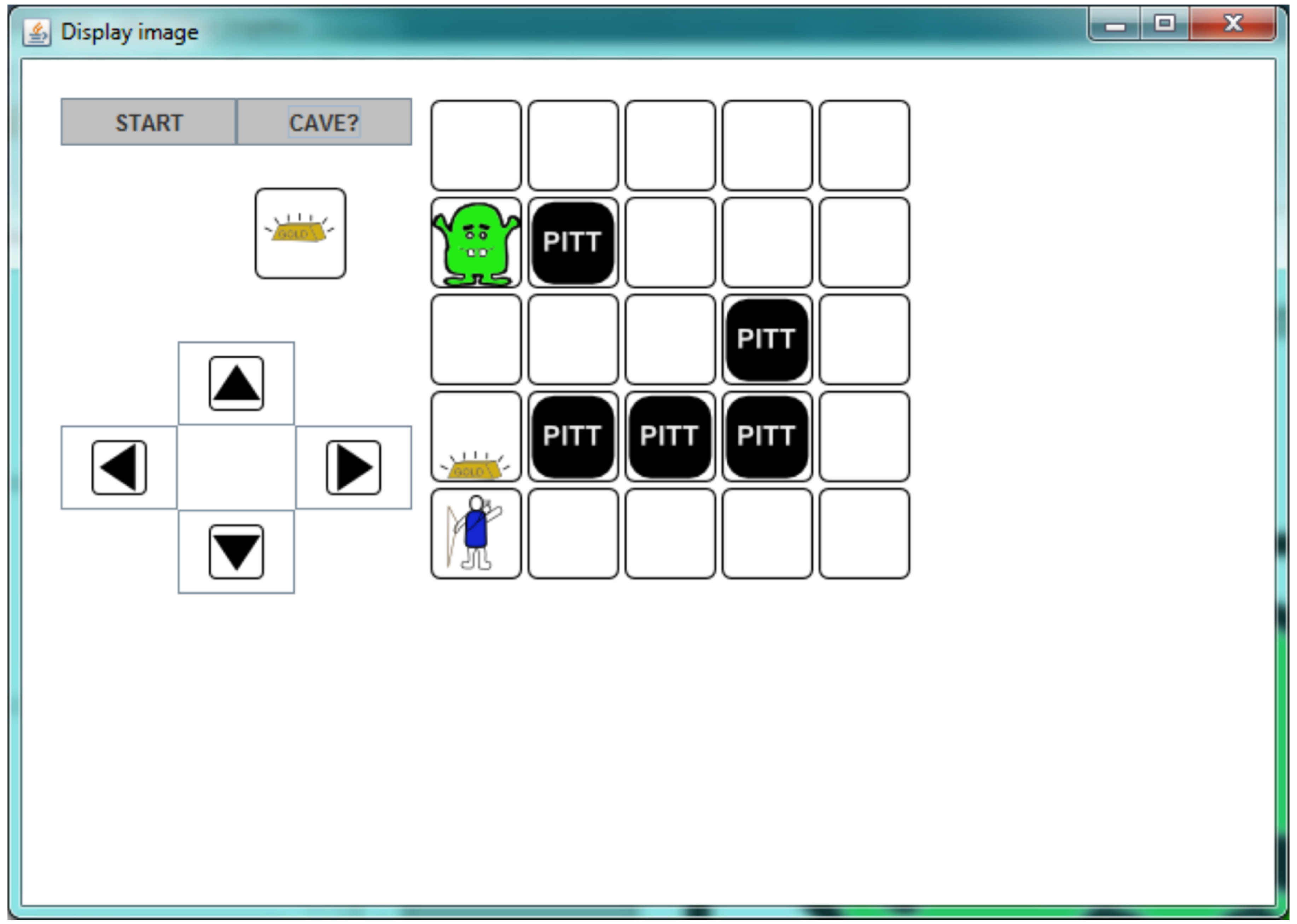
Wumpus



Wumpus

- Performance measure
  - –gold: +1000, death: -1000
  - –-1 per step, -10 for using the arrow

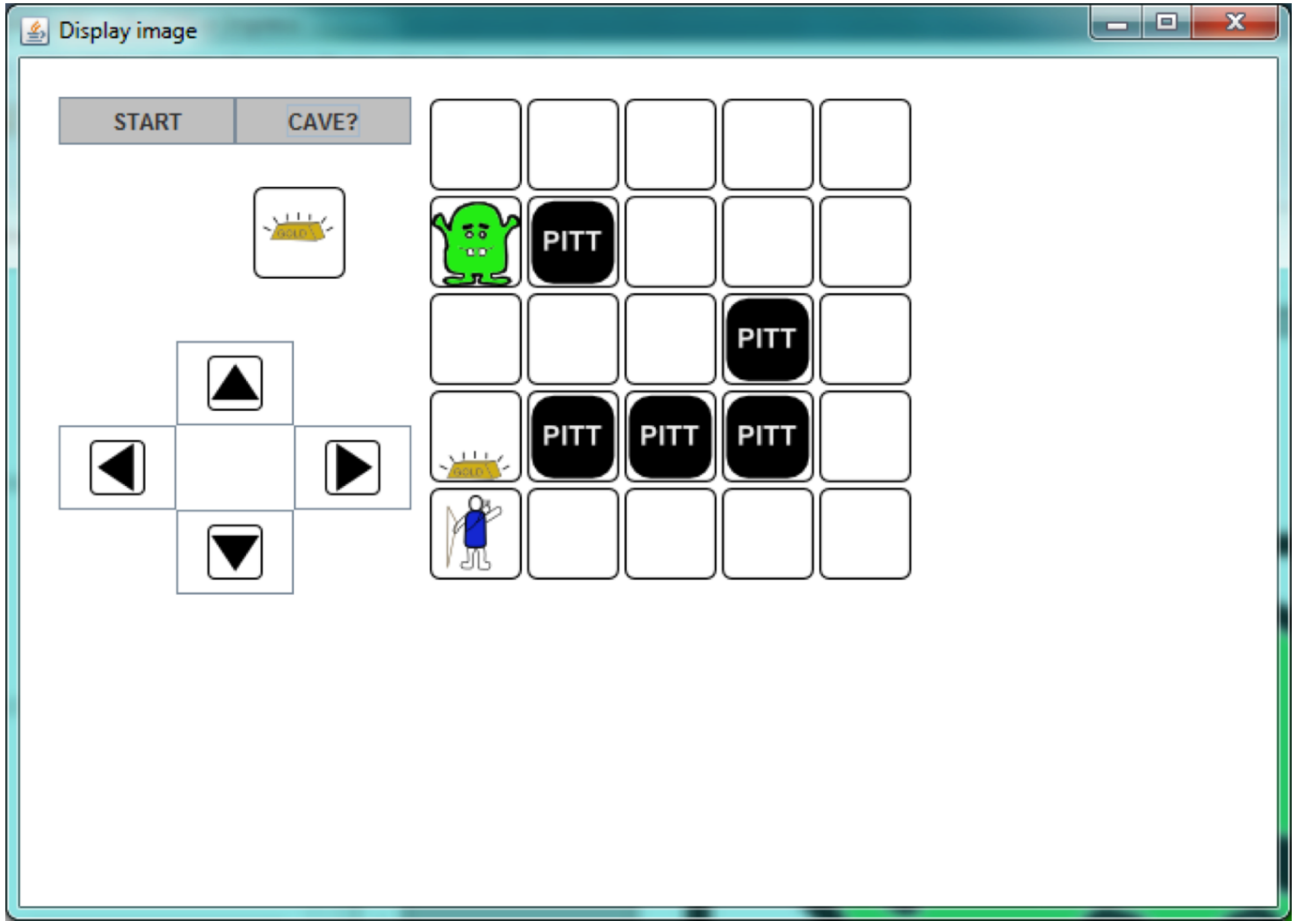
- Environment
  - –Squares adjacent to wumpus are smelly
  - –Squares adjacent to pit are breezy
  - –Glitter if gold is in the same square
  - –Shooting kills wumpus if you are facing it
  - –Shooting uses up the only arrow
  - –Grabbing picks up gold if in same square
  - –Releasing drops the gold in same square



- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Wumpus

- Cell
  - int row
  - Int col
- Hero
  - int row
  - Int col
- Wumpus
  - int row
  - Int col
- Pitt
  - int row
  - Int col
- Gold
  - int row
  - Int col



Wumpus

```
rule "Move Up" agenda-group "Commands" when
    (($mc : MoveCommand( move == Move.MOVE_FORWARD ) and
     $h  : Hero(direction == Direction.UP)) or
    ($mc : MoveCommand( move == Move.MOVE_BACKWARD ) and
     $h  : Hero(direction == Direction.DOWN)))
     $c  : Cell(row == $h.row + 1, col == $h.col )
then
    retract ( $mc );
    modify( $h ) { row = $h.row + 1 };
    modify( $c ) { hidden = false };
end

rule "Move Down" agenda-group "Commands" when
    (($mc : MoveCommand( move == Move.MOVE_FORWARD ) and
     $h  : Hero(direction == Direction.DOWN)) or
    ($mc : MoveCommand( move == Move.MOVE_BACKWARD ) and
     $h  : Hero(direction == Direction.UP)))
     $c  : Cell(row == $h.row - 1, col == $h.col )
then
    retract ( $mc );
    modify( $h ) { row = $h.row - 1 };
    modify( $c ) { hidden = false };
end
```

Wumpus

```
rule "Direction.UP, Move.TURN_LEFT"  agenda-group "Commands"  when
    $h  : Hero( direction == Direction.UP)
    $mc : MoveCommand( move == Move.TURN_LEFT )
then
    retract ( $mc );
    modify( $h ) { direction = Direction.LEFT };
end

rule "Direction DOWN,  MOVE.TURN_LEFT" agenda-group "Commands" when
    $h  : Hero( direction == Direction.DOWN)
    $mc : MoveCommand( move == Move.TURN_LEFT )
then
    retract ( $mc );
    modify( $h ) { direction = Direction.RIGHT };
end
```

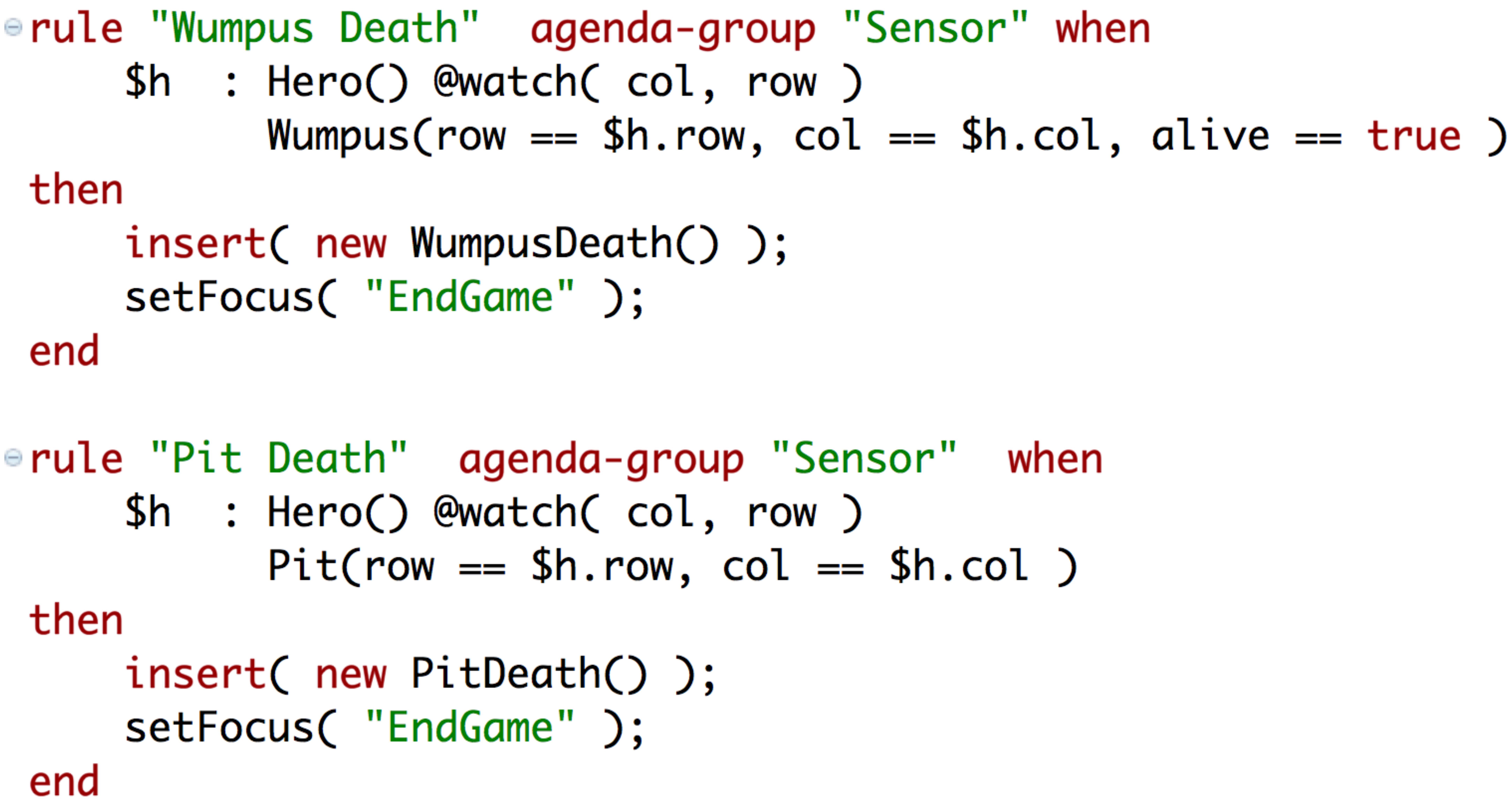
Wumpus

```
rule "Invalid Move"  agenda-group "Commands" when
     // Invalid Up
     ((($mc : MoveCommand( move == Move.MOVE_FORWARD ) and
     $h  : Hero(direction == Direction.UP)) or
     ($mc : MoveCommand( move == Move.MOVE_BACKWARD ) and
     $h  : Hero(direction == Direction.DOWN))) and
     not Cell(row == $h.row + 1, col == $h.col )) or

    // Invalid Down
     ((($mc : MoveCommand( move == Move.MOVE_FORWARD ) and
     $h  : Hero(direction == Direction.DOWN)) or
     ($mc : MoveCommand( move == Move.MOVE_BACKWARD ) and
     $h  : Hero(direction == Direction.UP))) and
     not Cell(row == $h.row - 1, col == $h.col )) or

     // Invalid LEFT
     ((($mc : MoveCommand( move == Move.MOVE_FORWARD ) and
     $h  : Hero(direction == Direction.LEFT)) or
     ($mc : MoveCommand( move == Move.MOVE_BACKWARD ) and
     $h  : Hero(direction == Direction.RIGHT))) and
     not Cell(row == $h.row, col == $h.col - 1 )) or

     // Invalid RIGHT
     ((($mc : MoveCommand( move == Move.MOVE_FORWARD ) and
     $h  : Hero(direction == Direction.RIGHT)) or
     ($mc : MoveCommand( move == Move.MOVE_BACKWARD ) and
     $h  : Hero(direction == Direction.LEFT))) and
     not Cell(row == $h.row, col == $h.col + 1 ) )
then
    retract( $mc );
    insert( new FeelBump() );
end
```

Wumpus

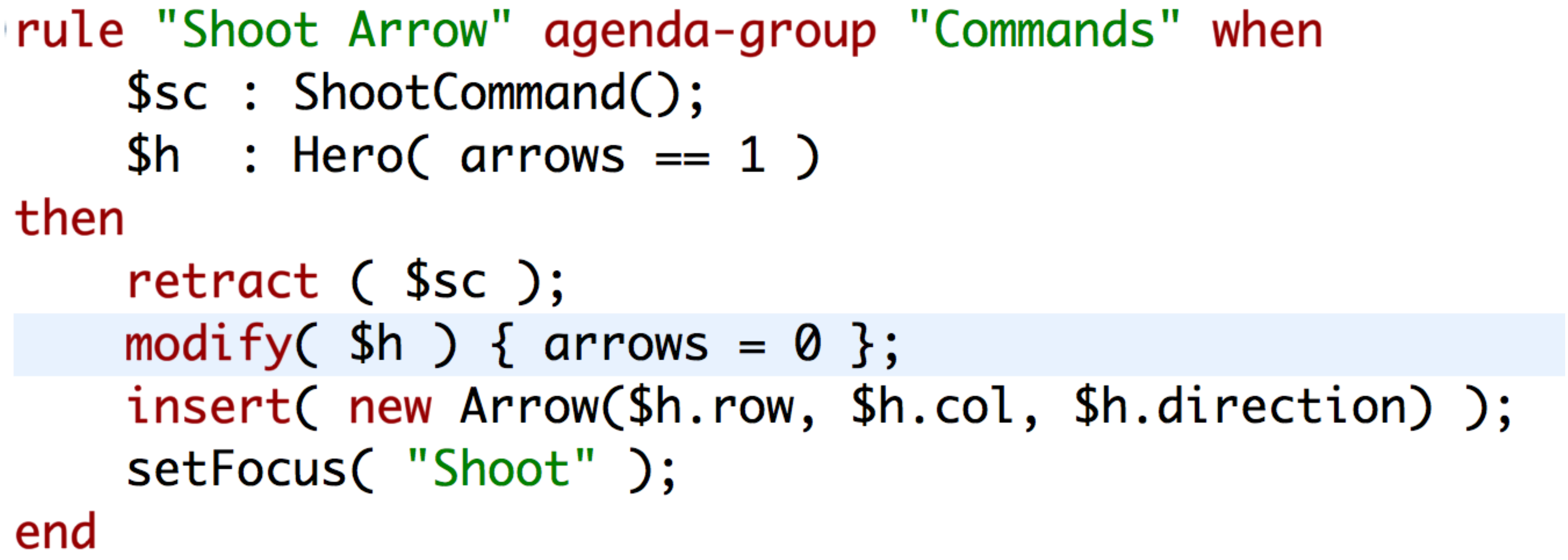
```
rule "Smell Stench" agenda-group "Sensor" when
    $h  : Hero() @watch( col, row )
          Wumpus(row == $h.row, col == $h.col ) or
          Wumpus(row == $h.row + 1, col == $h.col ) or
          Wumpus(row == $h.row - 1, col == $h.col ) or
          Wumpus(row == $h.row, col == $h.col + 1 ) or
          Wumpus(row == $h.row, col == $h.col - 1 )
then
    insertLogical( new SmellStench() );
end

rule "Feel Breeze"  agenda-group "Sensor"  when
    $h  : Hero() @watch( col, row )
          Pit(row == $h.row + 1, col == $h.col ) or
          Pit(row == $h.row - 1, col == $h.col ) or
          Pit(row == $h.row, col == $h.col + 1 ) or
          Pit(row == $h.row, col == $h.col - 1 )
then
    insertLogical( new FeelBreeze() );
end

rule "See Glitter"  agenda-group "Sensor"  when
    $h  : Hero( ) @watch( col, row )
          Gold(row == $h.row, col == $h.col )
then
    insertLogical( new SeeGlitter() );
end
```

Wumpus



```
rule "Wumpus Death"  agenda-group "Sensor" when
    $h  : Hero() @watch( col, row )
          Wumpus(row == $h.row, col == $h.col, alive == true )
then
    insert( new WumpusDeath() );
    setFocus( "EndGame" );
end

rule "Pit Death"  agenda-group "Sensor"  when
    $h  : Hero() @watch( col, row )
          Pit(row == $h.row, col == $h.col )
then
    insert( new PitDeath() );
    setFocus( "EndGame" );
end
```

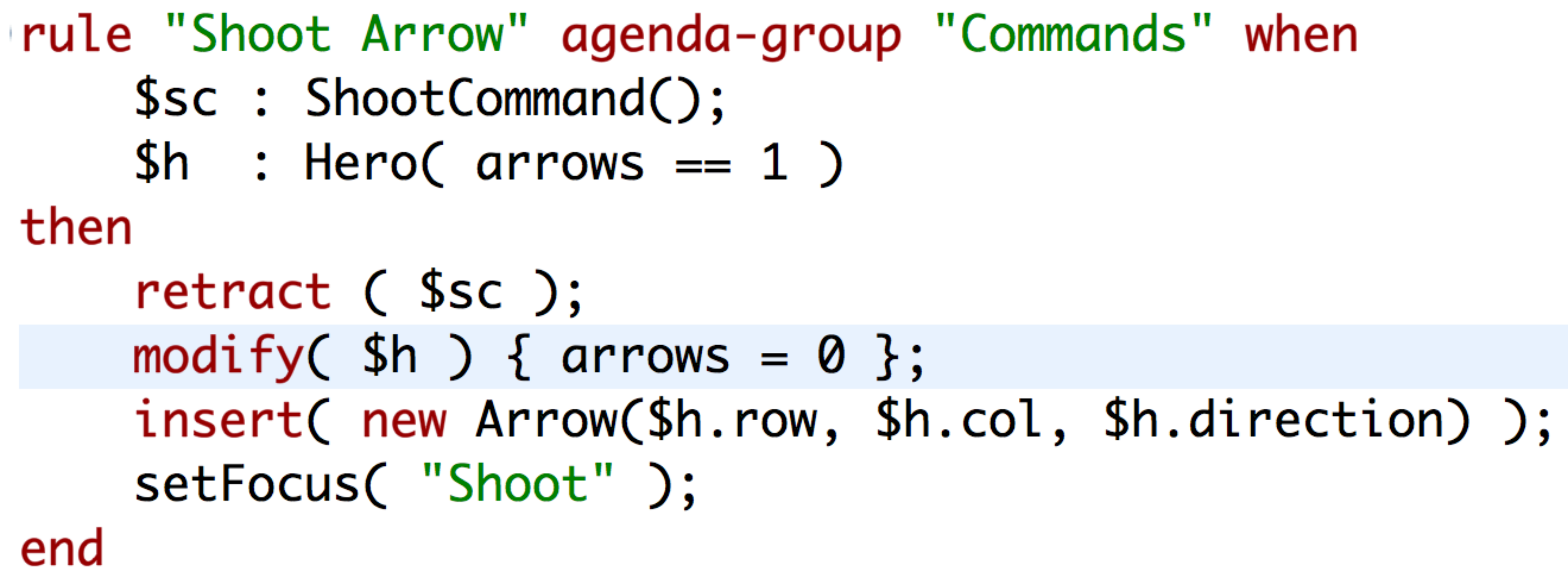
Wumpus



```
rule "Shoot Arrow" agenda-group "Commands" when
    $sc : ShootCommand();
    $h  : Hero( arrows == 1 )
then
    retract ( $sc );
    modify( $h ) { arrows = 0 };
    insert( new Arrow($h.row, $h.col, $h.direction) );
    setFocus( "Shoot" );
end
```

Wumpus

```
rule "Move Arrow Up" agenda-group "Shoot"  when
     $a :  Arrow( direction == Direction.UP)
then
    modify( $a ) { row = $a.row + 1 };
end

rule "Move Arrow Down" agenda-group "Shoot"  when
     $a :  Arrow( direction == Direction.DOWN)
then
    modify( $a ) { row = $a.row - 1 };
end

rule "Move Arrow Left" agenda-group "Shoot"  when
     $a :  Arrow( direction == Direction.LEFT)
then
    modify( $a ) { col = $a.col - 1 };
end
```

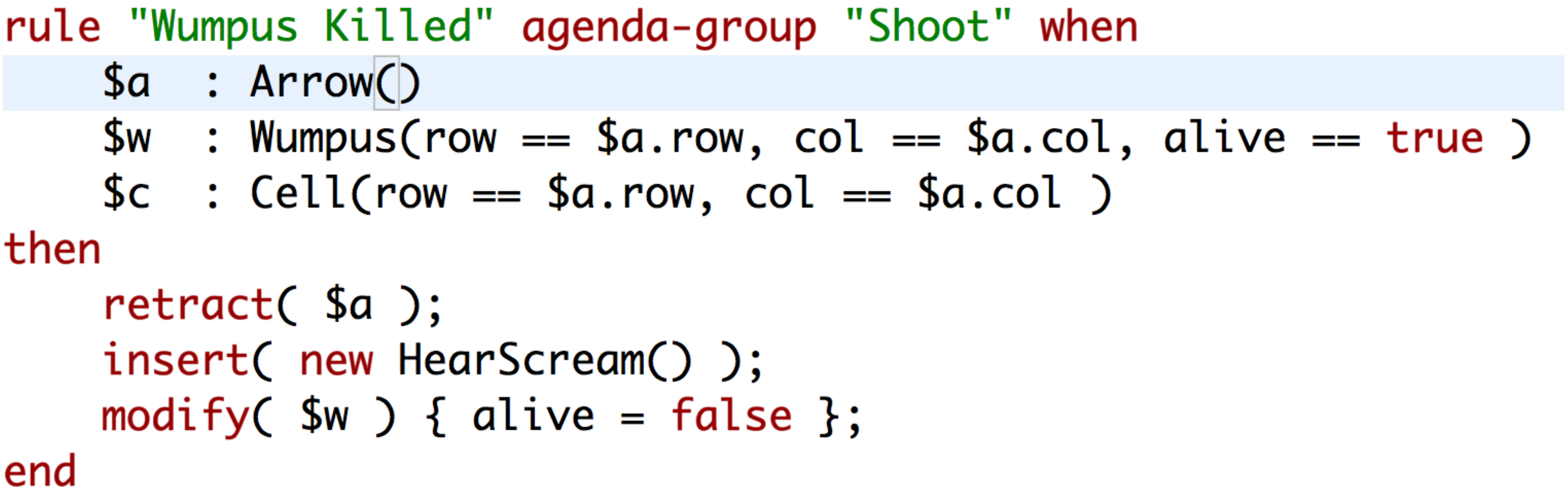


```
rule "Shoot Arrow" agenda-group "Commands" when
    $sc : ShootCommand();
    $h  : Hero( arrows == 1 )
then
    retract ( $sc );
    modify( $h ) { arrows = 0 };
    insert( new Arrow($h.row, $h.col, $h.direction) );
    setFocus( "Shoot" );
end
```

Wumpus

```
rule "Cave Boundary, Remove Arrow" agenda-group "Shoot"  when
     $a : Arrow()
     not Cell(row == $a.row, col == $a.col )
then
    retract ( $a );
end
```

Wumpus



```
rule "Wumpus Killed" agenda-group "Shoot" when
    $a  : Arrow()
    $w  : Wumpus(row == $a.row, col == $a.col, alive == true )
    $c  : Cell(row == $a.row, col == $a.col )
then
    retract( $a );
    insert( new HearScream() );
    modify( $w ) { alive = false };
end
```

Adventures in Drools

Adventures

```
rooms = [
    "basement"             : new Room("basement"),
    "lounge"               : new Room("lounge"),
    "dining room”          : new Room("dining room"),
    “kitchen"              : new Room("kitchen"),
    "ground floor hallway" : new Room("ground floor hallway"),
    "bedroom1"             : new Room("bedroom1"),
    "bedroom2"             : new Room("bedroom2"),
    “bathroom"             : new Room("bathroom"),
    "office"               : new Room("office"),
    "first floor hallway"  : new Room("first floor hallway")
];
```

```
doors = [
    "d1" : new Door( rooms[“kitchen",               rooms["basement"] ),

    "d2" : new Door( rooms["ground floor hallway"], rooms["lounge"]),
    "d3" : new Door( rooms["ground floor hallway"], rooms["dining room"] ),
    "d4" : new Door( rooms["ground floor hallway"], rooms["kitchen"]),
    "d5" : new Door( rooms["ground floor hallway"], rooms[ "first floor hallway"] ),

    "d6" : new Door( rooms["first floor hallway"],  rooms[ "bedroom1"] ),
    "d7" : new Door( rooms["first floor hallway”],  rooms[ "bedroom2"] ),
    "d8" : new Door( rooms["first floor hallway"],  rooms[ "bathroom"] ),
    "d9" : new Door( rooms["first floor hallway”],  rooms[ "office"] )
];
```

Adventures

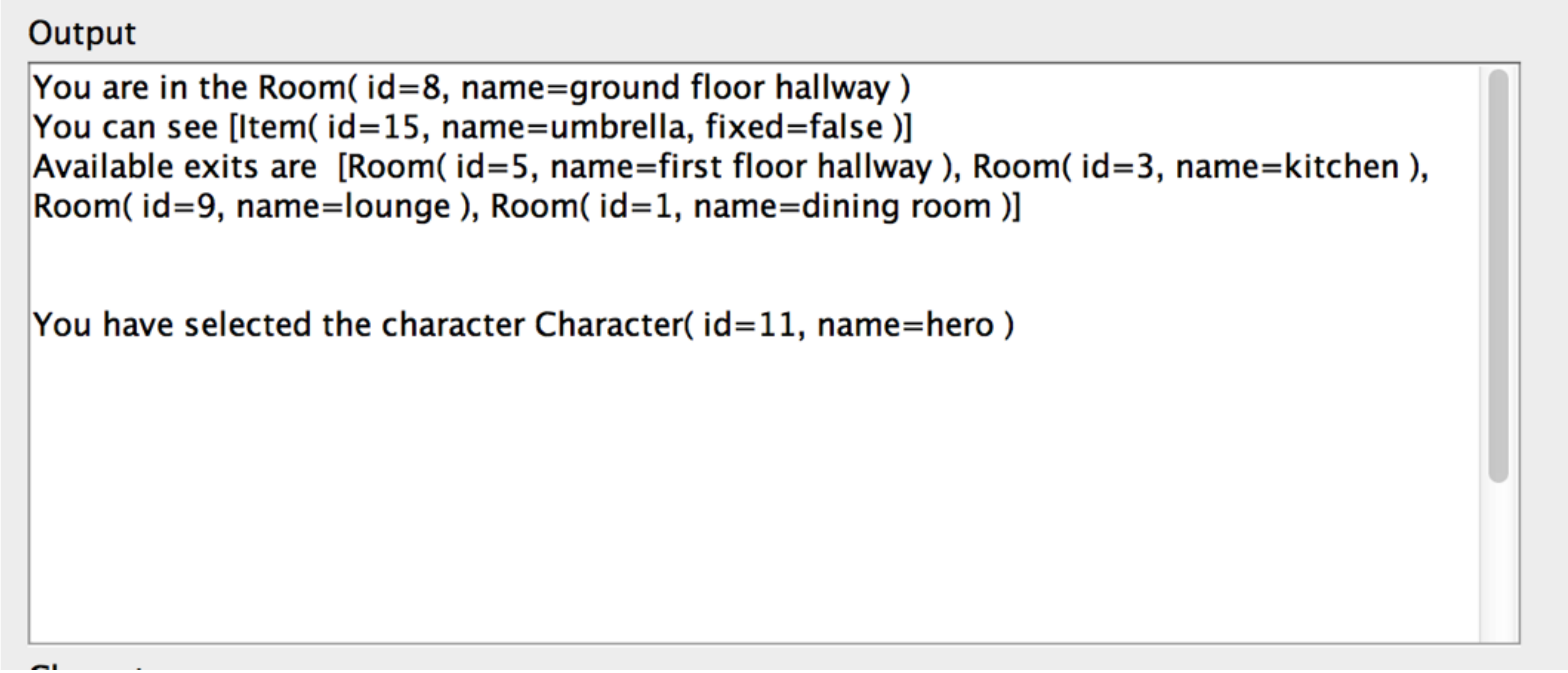
```
characters = [ "hero" : new Character( "hero" ),
               "monster" : new Character( "monster" ) ];
```

```
items = [
    "umbrella" : new Item( "umbrella" ),
    "desk" : new Item( "desk", false ),
    "draw" : new Item( "draw", false ),
    "envelop" : new Item( "envelop" ),
    "key1" : new Key("basement key")
];
```

```
with(doors["d1"]){ lockStatus =  LockStatus.LOCKED, key = items["key1"] };
```

```
locations = [
    "monster" :  new Location( characters["monster"], rooms["basement"] ),
    "hero" :  new Location( characters["hero"], rooms["ground floor hallway"] ),
    "umbrella" :  new Location( items["umbrella"], rooms["lounge"] ),
    "desk" :  new Location( items["desk"], rooms["office"] ),
    "draw" :  new Location( items["draw"], items["desk"] ),
    "envelop" :  new Location( items["envelop"], items["draw"] ),
    "key1" :  new Location( items["key1"], items["envelop"] )
];
```

Adventures



```
You are in the Room( id=8, name=ground floor hallway )
You can see [Item( id=15, name=umbrella, fixed=false )]
Available exits are  [Room( id=5, name=first floor hallway ), Room( id=3, name=kitchen ),
Room( id=9, name=lounge ), Room( id=1, name=dining room )]


You have selected the character Character( id=11, name=hero )
```

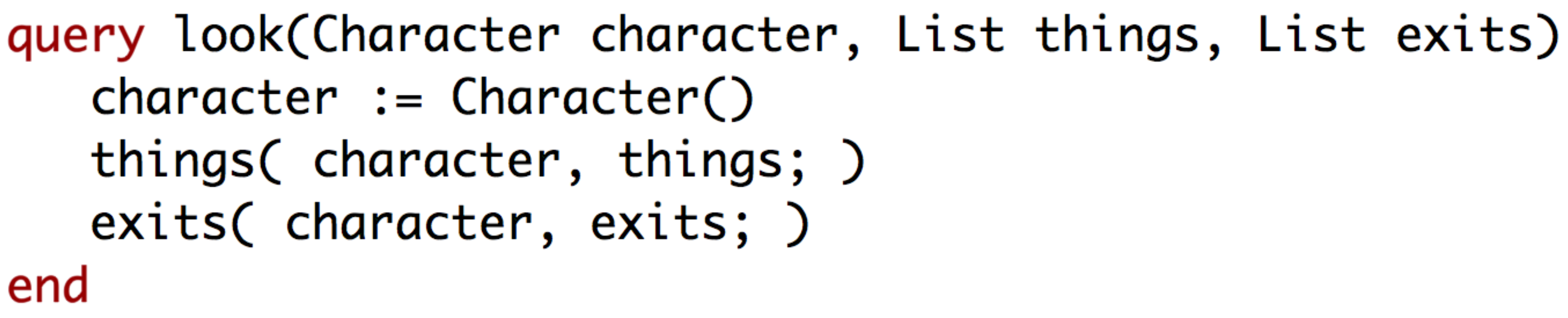
```
rule Look agenda-group "commands"  when
    lc : LookCommand(  c : character )
    l : Location( thing == c, )
    ?look( c, things, exits; )
then
    str = "You are in the " + l.room.name + "\n";
    str +="You can see " + thingsToString( things ) + "\n";
    str +="Available exits are  " + thingsToString( exits ) + "\n";
    str +="\n";

    lc.session.channels["output"].send( str );
end
```

Adventures

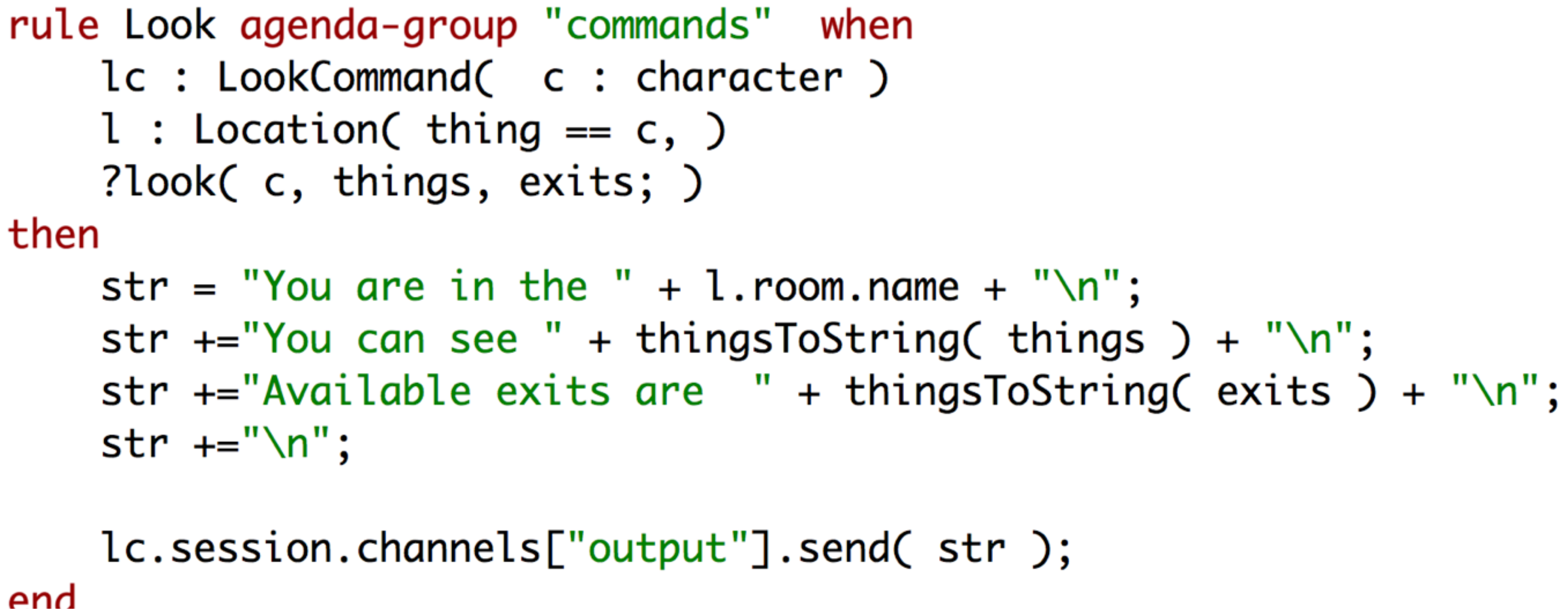
```
rule Look agenda-group "commands"  when
    lc : LookCommand(  c : character )
    l : Location( thing == c, )
    ?look( c, things, exits; )
then
    str = "You are in the " + l.room.name + "\n";
    str +="You can see " + thingsToString( things ) + "\n";
    str +="Available exits are  " + thingsToString( exits ) + "\n";
    str +="\n";

    lc.session.channels["output"].send( str );
end
```

```
query look(Character character, List things, List exits)
    character := Character()
    things( character, things; )
    exits( character, exits; )
end
```

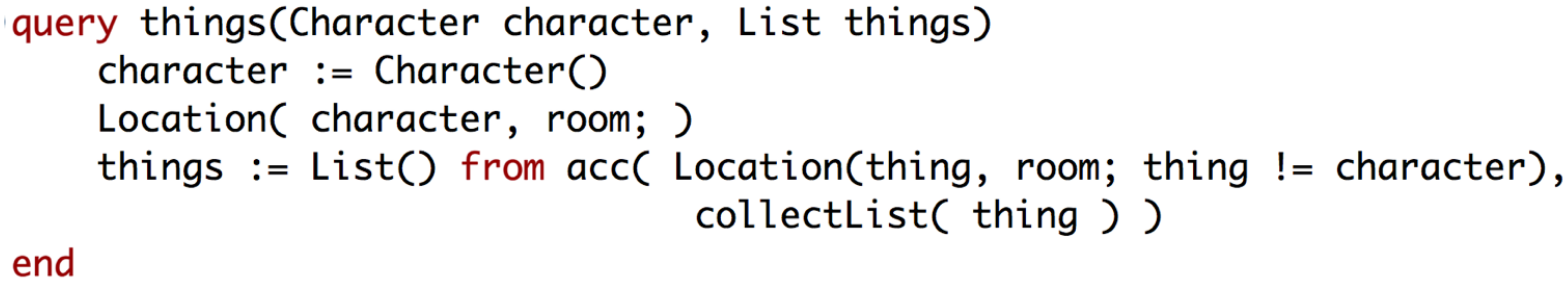
Adventures



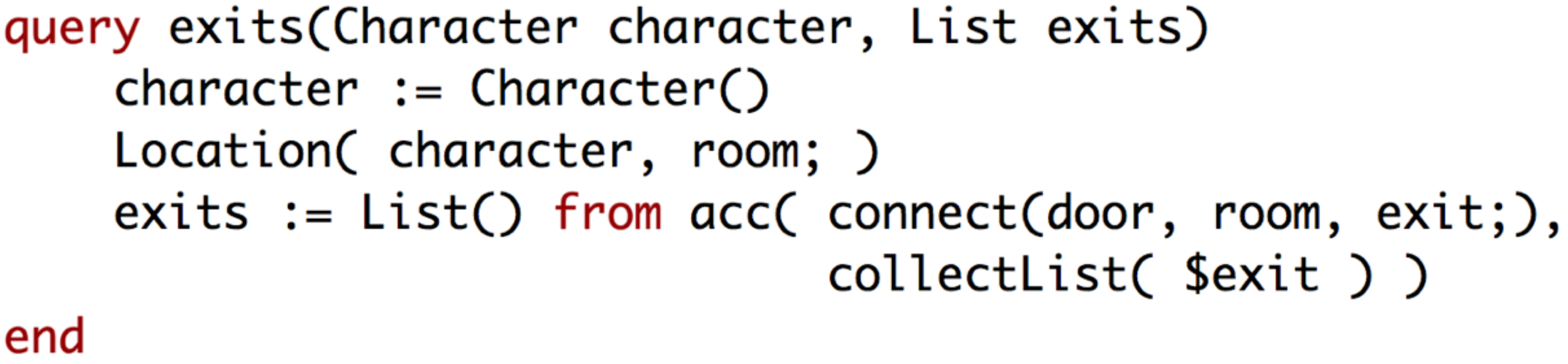
```
query look(Character character, List things, List exits)
   character := Character()
   things( character, things; )
   exits( character, exits; )
end
```



```
rule Look agenda-group "commands"  when
    lc : LookCommand(  c : character )
    l : Location( thing == c, )
    ?look( c, things, exits; )
then
    str = "You are in the " + l.room.name + "\n";
    str +="You can see " + thingsToString( things ) + "\n";
    str +="Available exits are  " + thingsToString( exits ) + "\n";
    str +="\n";

    lc.session.channels["output"].send( str );
end
```

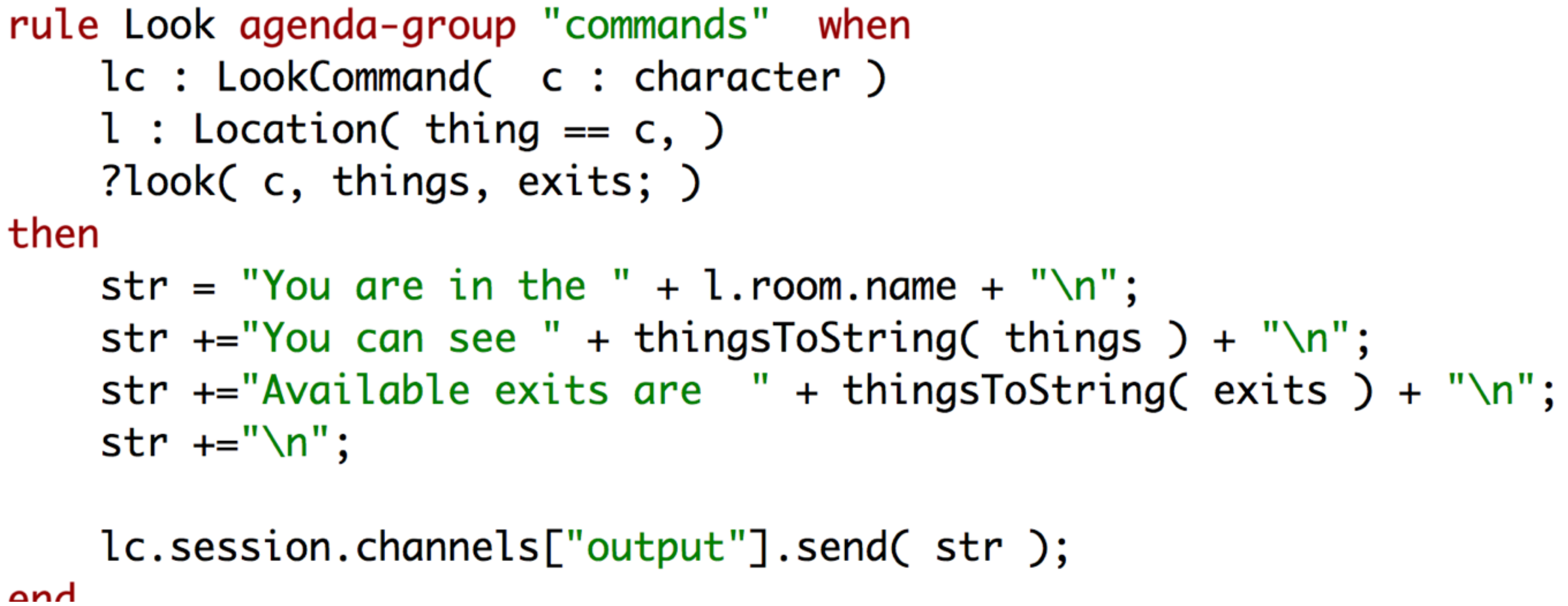


```
query things(Character character, List things)
    character := Character()
    Location( character, room; )
    things := List() from acc( Location(thing, room; thing != character),
                               collectList( thing ) )
end
```

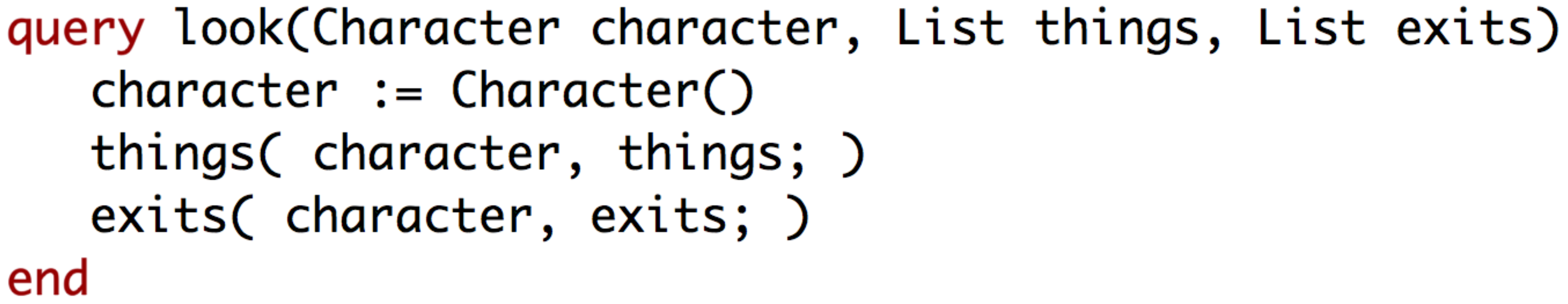


```
query exits(Character character, List exits)
    character := Character()
    Location( character, room; )
    exits := List() from acc( connect(door, room, exit;),
                              collectList( $exit ) )
end
```

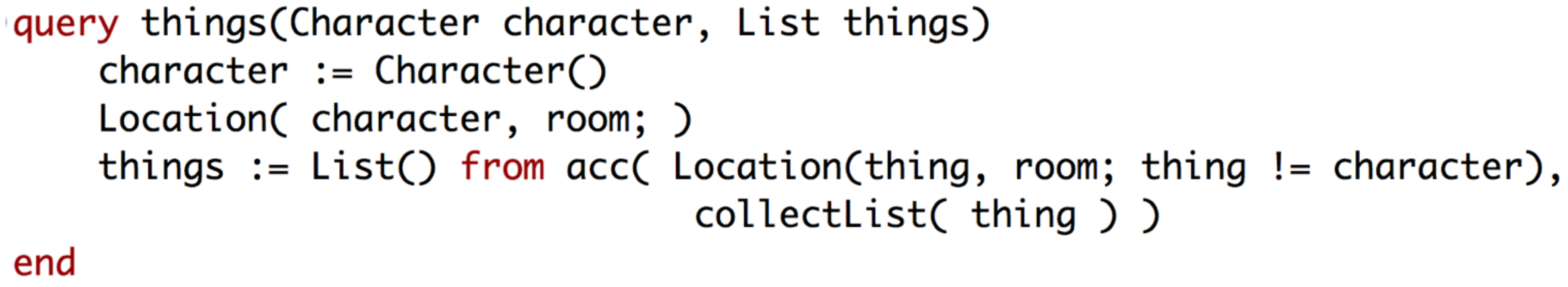
Adventures



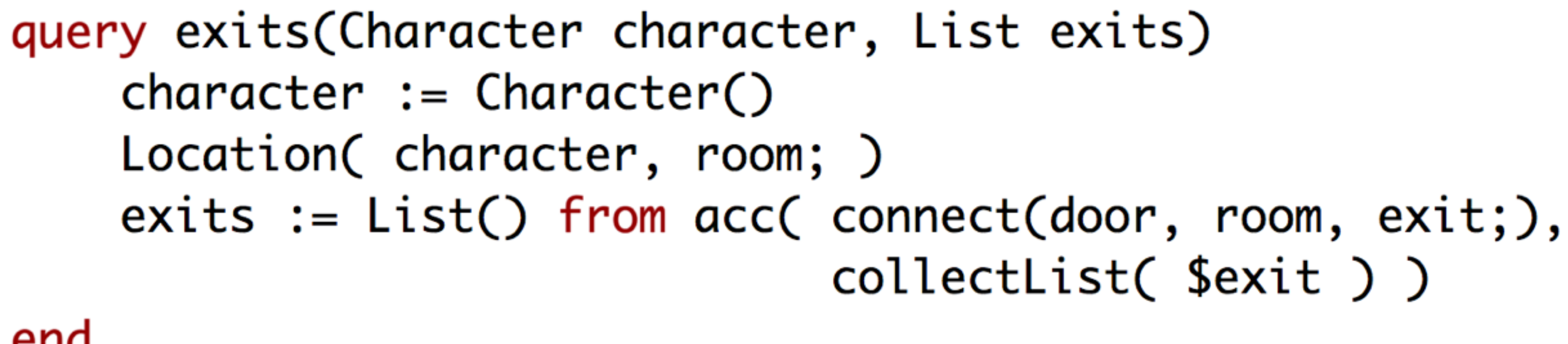
```
rule Look agenda-group "commands"  when
    lc : LookCommand(  c : character )
    l : Location( thing == c, )
    ?look( c, things, exits; )
then
    str = "You are in the " + l.room.name + "\n";
    str +="You can see " + thingsToString( things ) + "\n";
    str +="Available exits are  " + thingsToString( exits ) + "\n";
    str +="\n";

    lc.session.channels["output"].send( str );
end
```



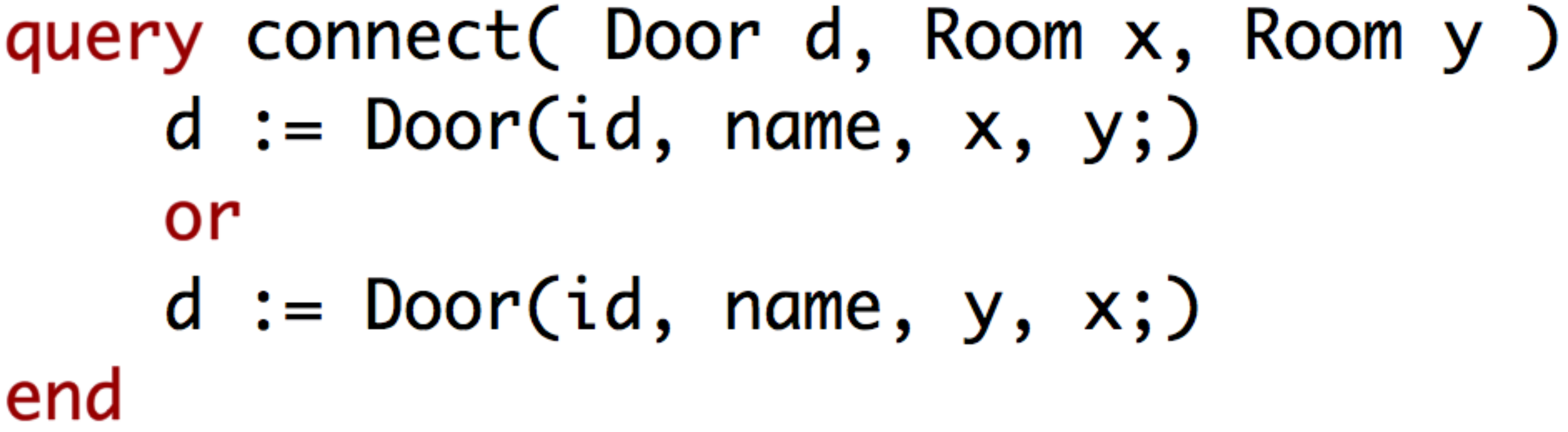
```
query look(Character character, List things, List exits)
   character := Character()
   things( character, things; )
   exits( character, exits; )
end
```



```
query things(Character character, List things)
    character := Character()
    Location( character, room; )
    things := List() from acc( Location(thing, room; thing != character),
                               collectList( thing ) )
end
```



```
query exits(Character character, List exits)
    character := Character()
    Location( character, room; )
    exits := List() from acc( connect(door, room, exit;),
                              collectList( $exit ) )
end
```

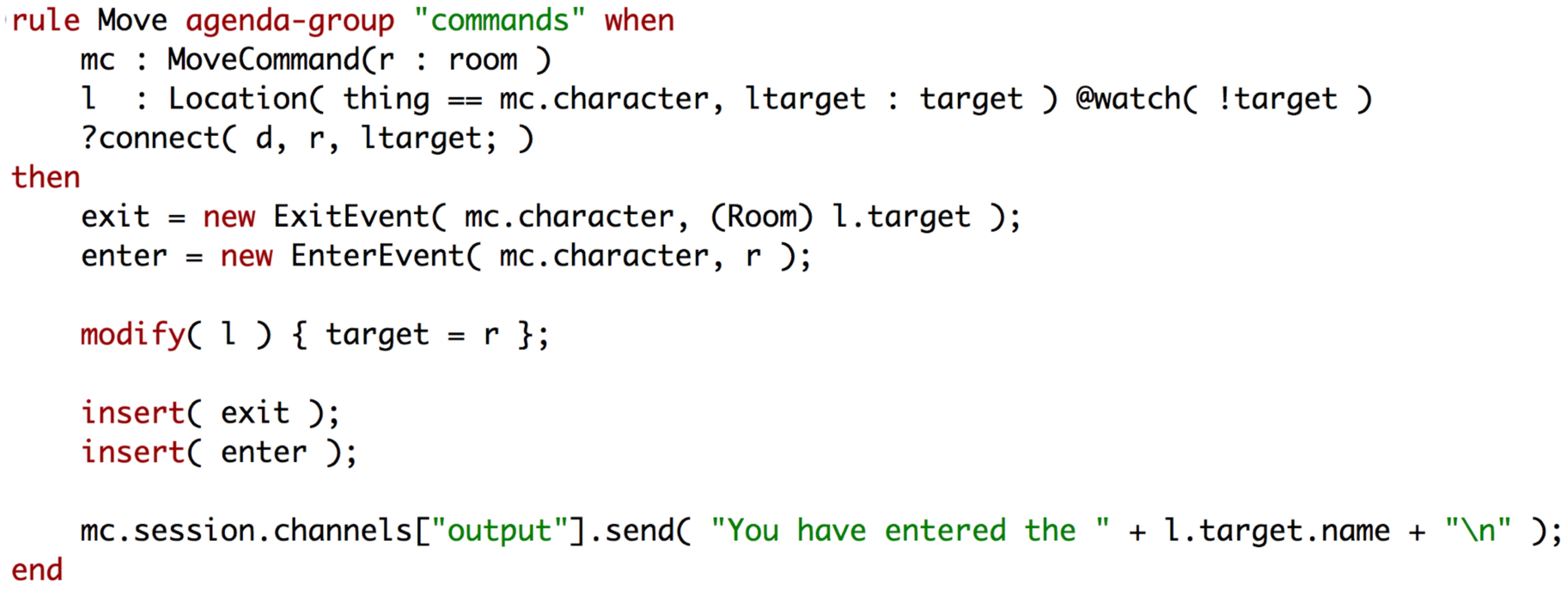


```
query connect( Door d, Room x, Room y )
    d := Door(id, name, x, y;)
    or
    d := Door(id, name, y, x;)
end
```

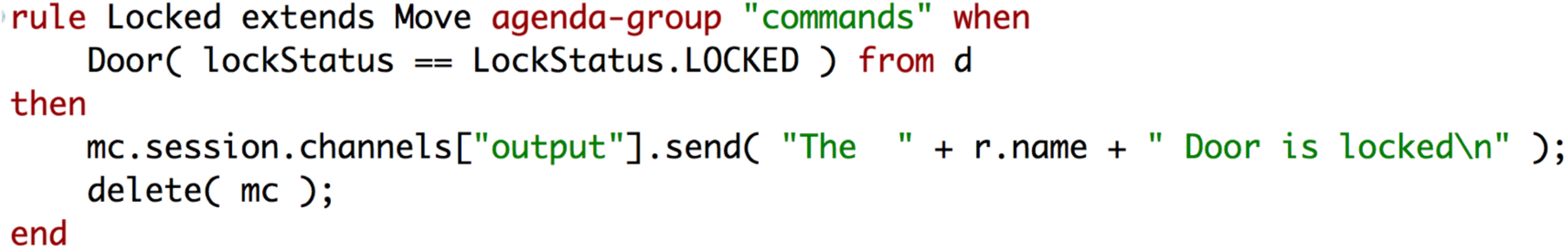
Adventures

```
rule Move agenda-group "commands" when
    mc : MoveCommand(r : room )
    l  : Location( thing == mc.character, ltarget : target ) @watch( !target )
    ?connect( d, r, ltarget; )
then
    exit = new ExitEvent( mc.character, (Room) l.target );
    enter = new EnterEvent( mc.character, r );

    modify( l ) { target = r };

    insert( exit );
    insert( enter );

    mc.session.channels["output"].send( "You have entered the " + l.target.name + "\n" )
end
```

Adventures



```
rule Move agenda-group "commands" when
    mc : MoveCommand(r : room )
    l  : Location( thing == mc.character, ltarget : target ) @watch( !target )
    ?connect( d, r, ltarget; )
then
    exit = new ExitEvent( mc.character, (Room) l.target );
    enter = new EnterEvent( mc.character, r );

    modify( l ) { target = r };

    insert( exit );
    insert( enter );

    mc.session.channels["output"].send( "You have entered the " + l.target.name + "\n" );
end
```

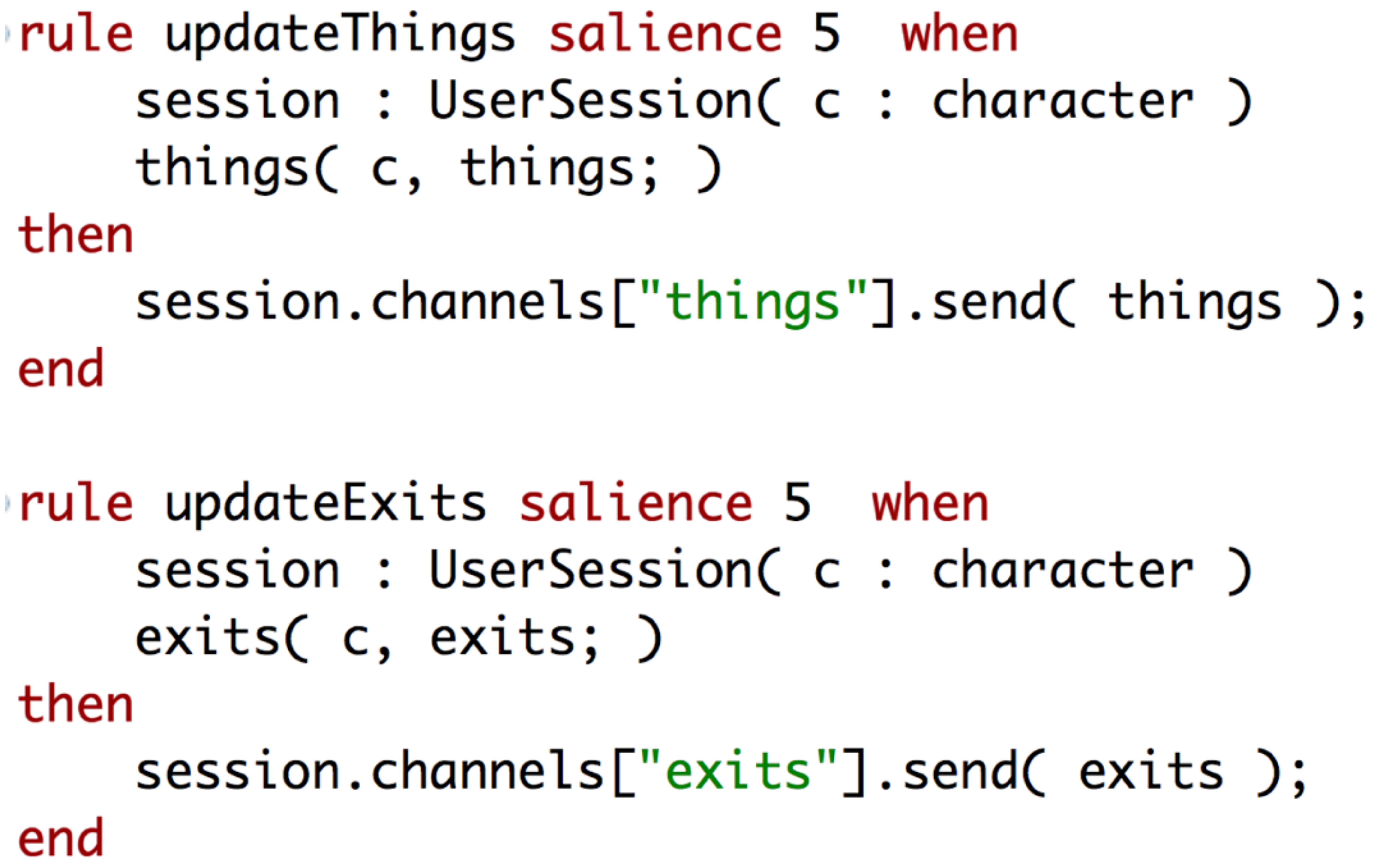


```
rule Locked extends Move agenda-group "commands" when
    Door( lockStatus == LockStatus.LOCKED ) from d
then
    mc.session.channels["output"].send( "The  " + r.name + " Door is locked\n" );
    delete( mc );
end
```

Adventures

```
rule UnlockingDoors agenda-group "commands"  when
    uc : UseCommand()
    r : Room() from uc.target
    cl  : Location( thing == uc.character, ltarget : target  )
    ?connect( door, ltarget, r; )
    if( door.key != uc.thing) break[wrongKey]
    if( door.lockStatus == LockStatus.UNLOCKED) break[alreadyUnlocked]
then
    modify(door){ lockStatus = LockStatus.UNLOCKED };
    uc.session.channels["output"].send(  "You have unlocked the " + r.name + " door\n" );
    retract ( uc );
then[wrongKey]
    uc.session.channels["output"].send(  "The selected key cannot open the " + r.name + " door\n" );
    retract ( uc );
then[alreadyUnlocked]
    uc.session.channels["output"].send(  "The " + r.name + " door is already unlocked\n" );
    retract ( uc );
end
```

Adventures



```
rule updateThings salience 5  when
    session : UserSession( c : character )
    things( c, things; )
then
    session.channels["things"].send( things );
end

rule updateExits salience 5  when
    session : UserSession( c : character )
    exits( c, exits; )
then
    session.channels["exits"].send( exits );
end
```

Adventures

```
rule updateThings salience 5  when
    session : UserSession( c : character )
    things( c, things; )
then
    session.channels["things"].send( things );
end

rule updateExits salience 5  when
    session : UserSession( c : character )
    exits( c, exits; )
then
    session.channels["exits"].send( exits );
end
```

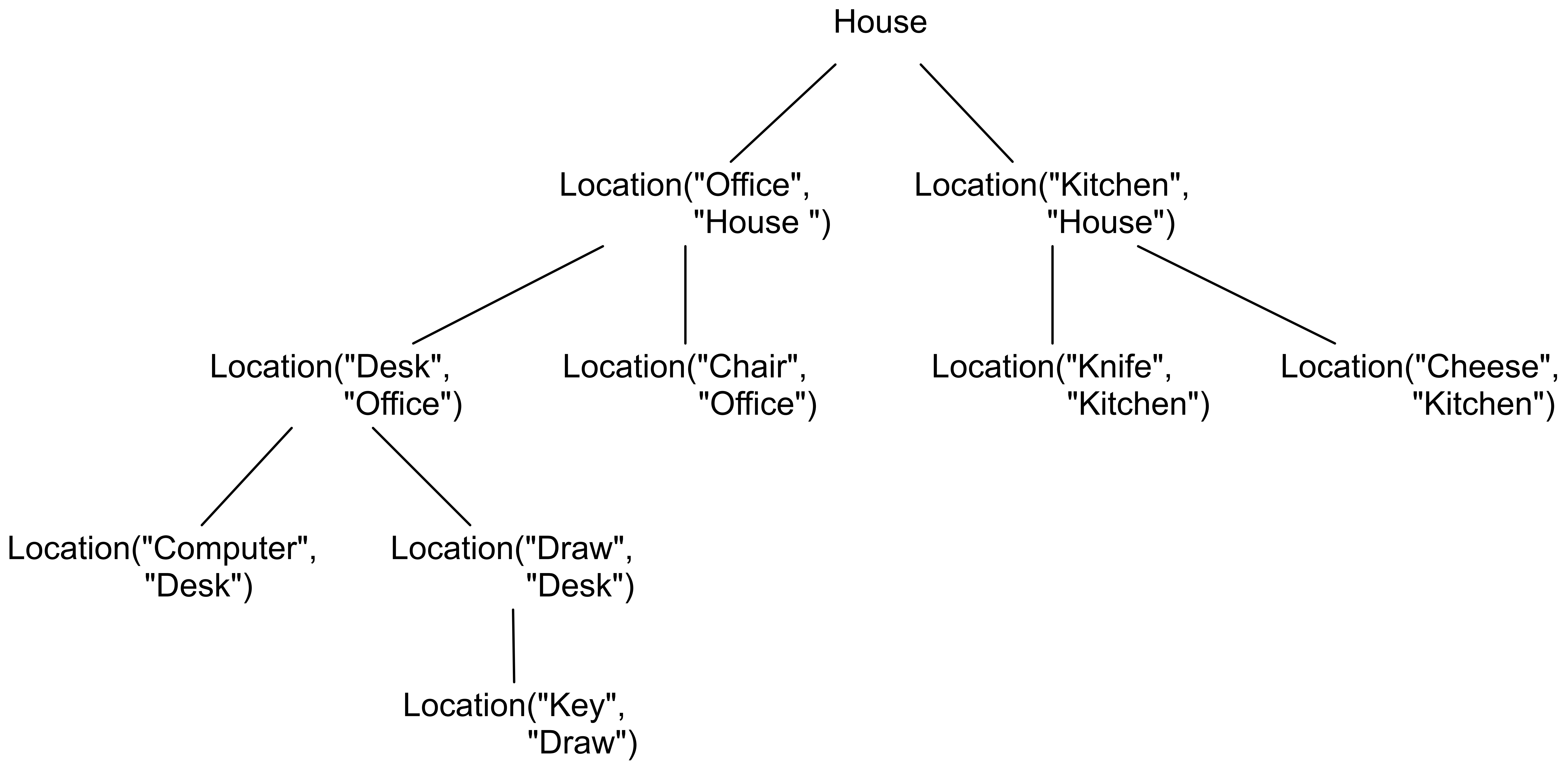
```
query things(Character character, List things)
    character := Character()
    Location( character, room; )
    things := List() from acc( Location(thing, room; thing != character),
                               collectList( thing ) )
end

query exits(Character character, List exits)
    character := Character()
    Location( character, room; )
    exits := List() from acc( connect(door, room, exit;),
                              collectList( exit ) )
end
```

Adventures

```
"desk" :  new Location( items["desk"], rooms["office"] ),
"draw" :  new Location( items["draw"], items["desk"] ),
"envelop" :  new Location( items["envelop"], items["draw"] ),
"key1" :  new Location( items["key1"], items["envelop"] )
```

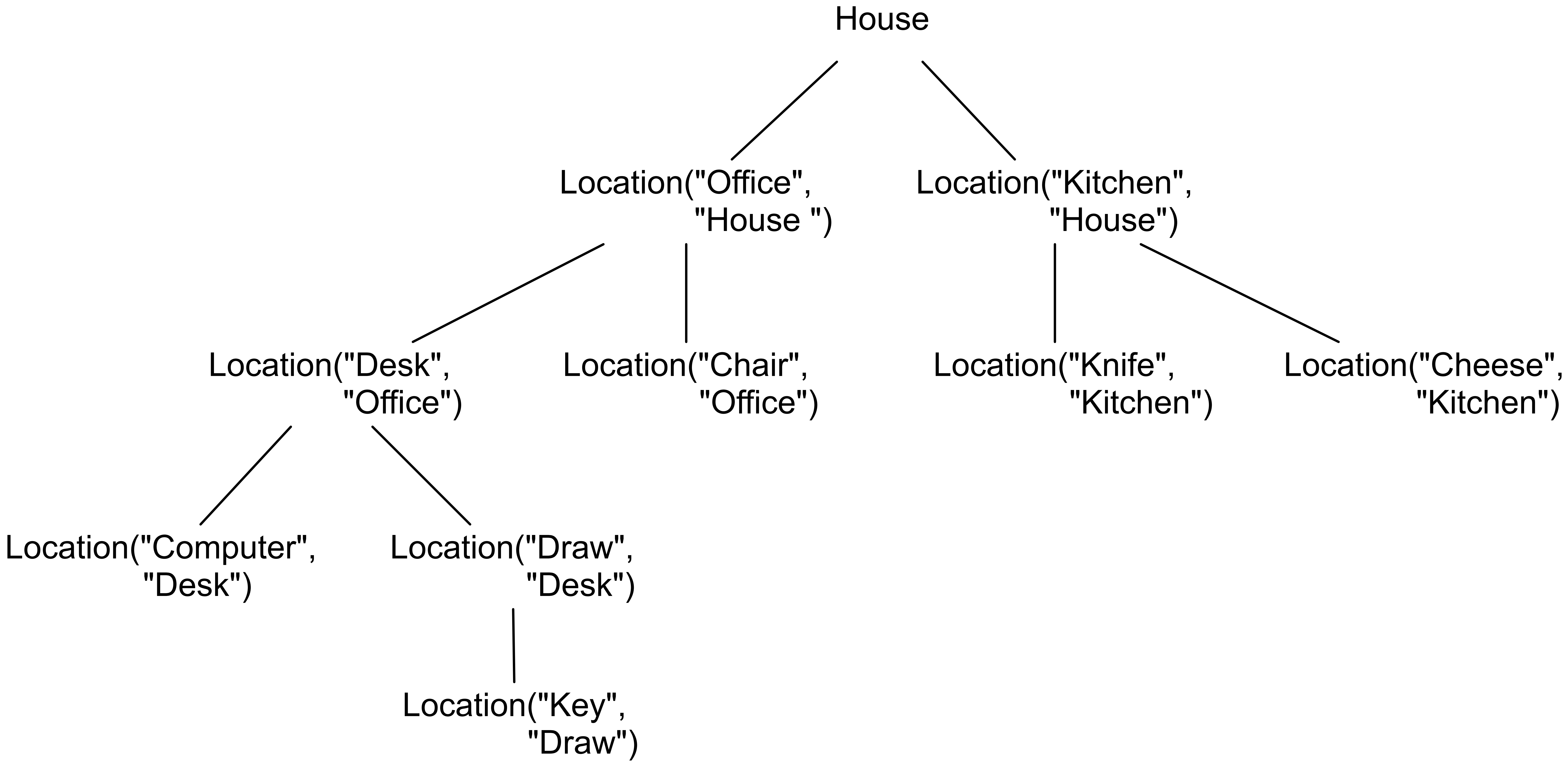
```
rule Search agenda-group "commands"  when
    sc : SearchCommand( t : thing, t != null )
    session : UserSession( )
    acc( ?isContainedIn(child, r, t;);
         strThings : collectList( child.name + " in " + r.name ),
         things : collectList( child ))
then
    sc.session.channels["output"].send( "found " + strThings + "\n" );
    session.channels["things"].send( things );
end
```

Reasoning with Graphs



Backward Chaining

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

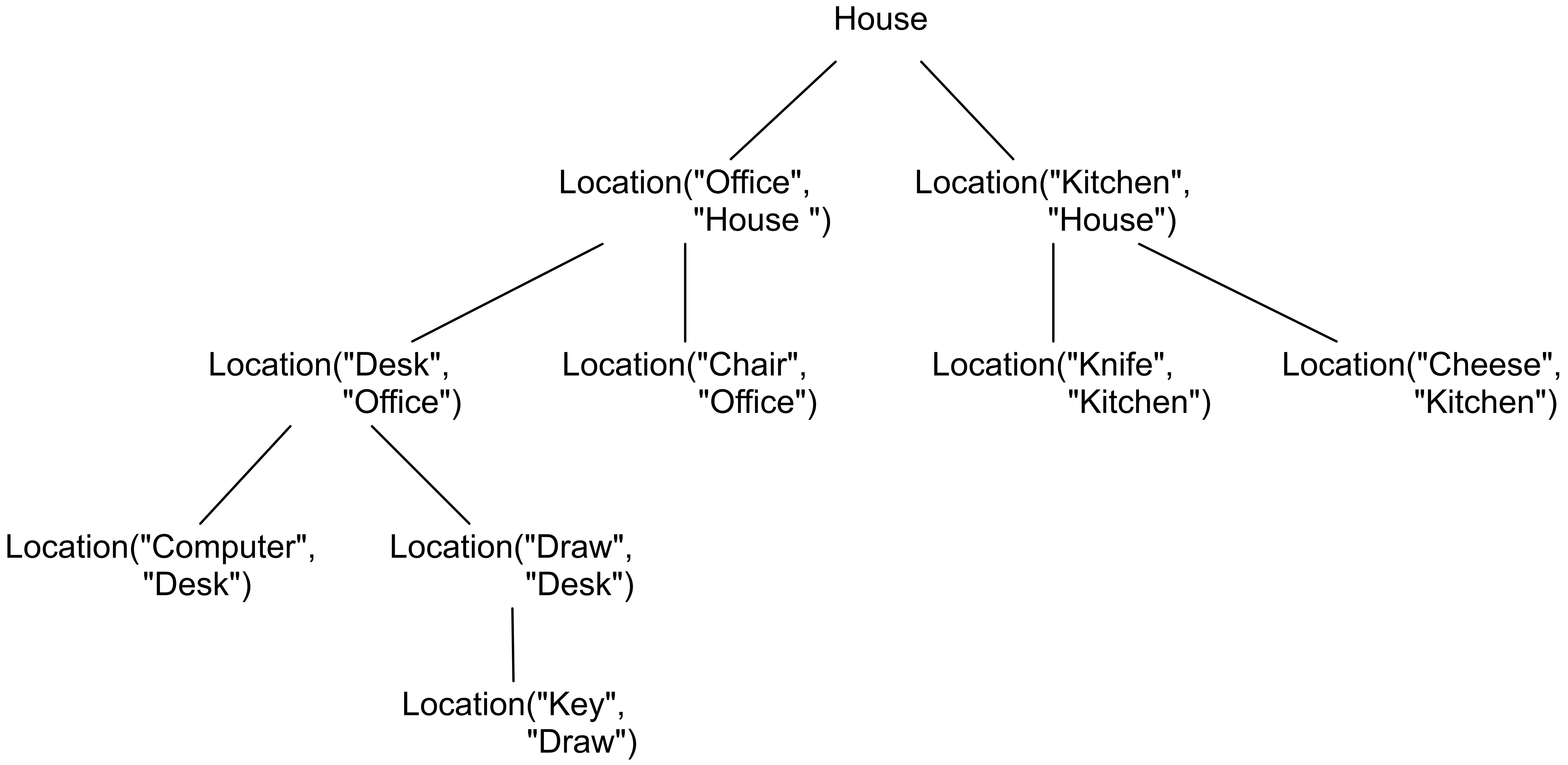
Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

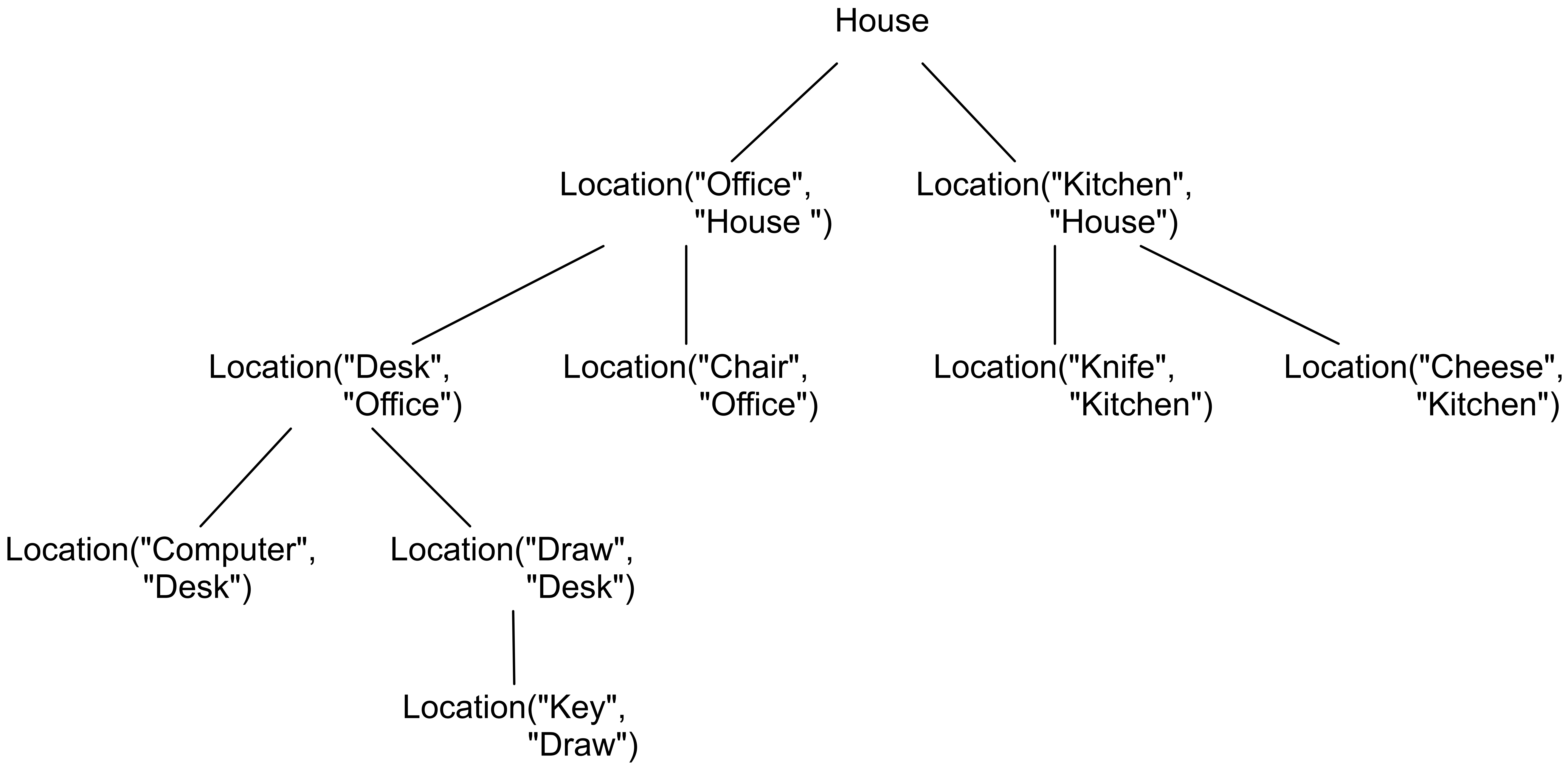
```
ksession.insert( new Location("Office", "House") );
ksession.insert( new Location("Kitchen", "House") );
ksession.insert( new Location("Knife", "Kitchen") );
ksession.insert( new Location("Cheese", "Kitchen") );
ksession.insert( new Location("Desk", "Office") );
ksession.insert( new Location("Chair", "Office") );
ksession.insert( new Location("Computer", "Desk") );
ksession.insert( new Location("Draw", "Desk") );
```



House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

Backward Chaining

```
rule "go" salience 10
when
    $s : String(  )
then
    System.out.println( $s );
end
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

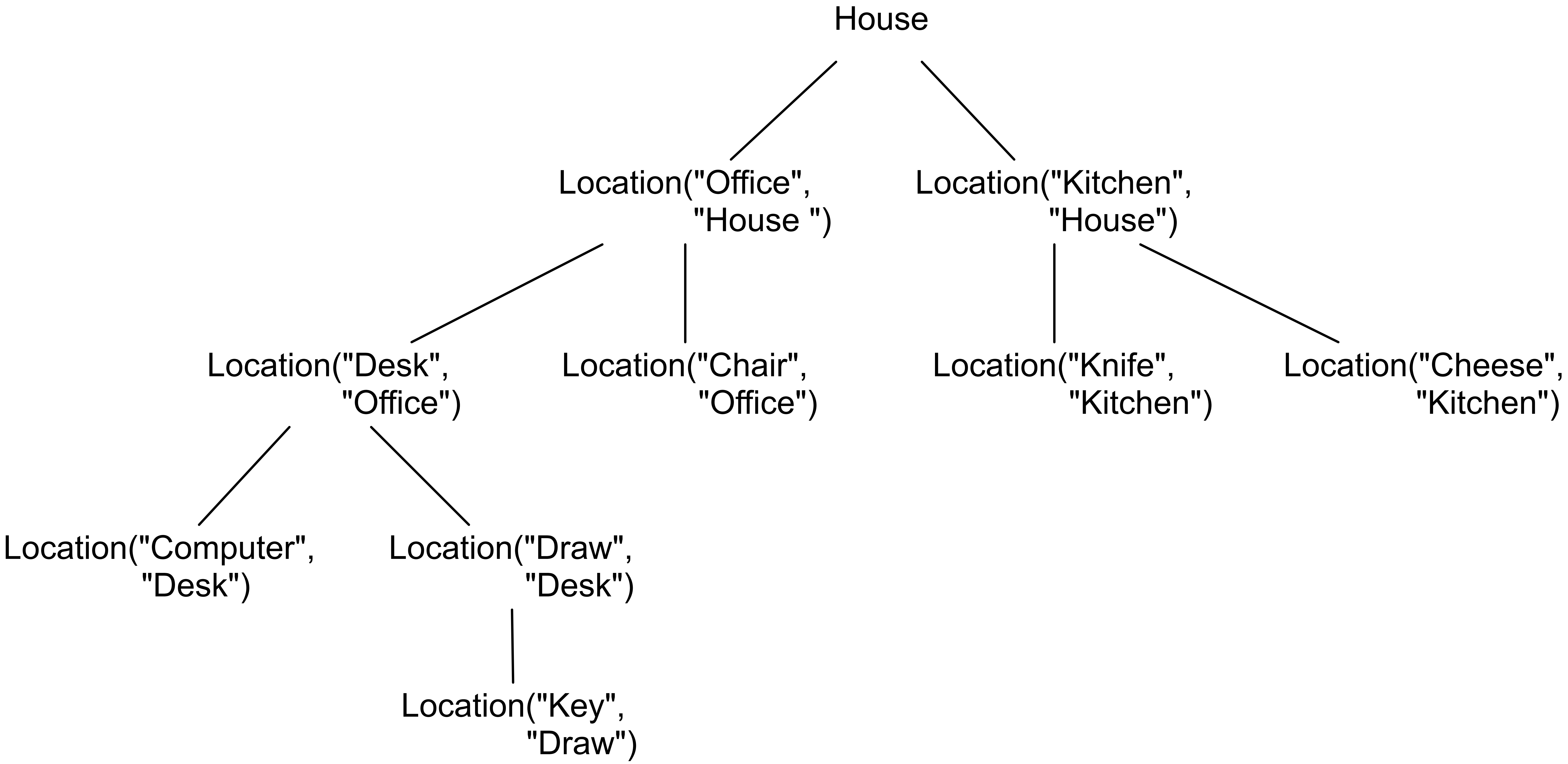
Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

```
rule "go" salience 10
when
    $s : String(  )
then
    System.out.println( $s );
end
```

```
rule "go1"
when
    String( this == "go1" )
    isContainedIn("Office", "House"; )
then
    System.out.println( "office is in the house" );
end
```

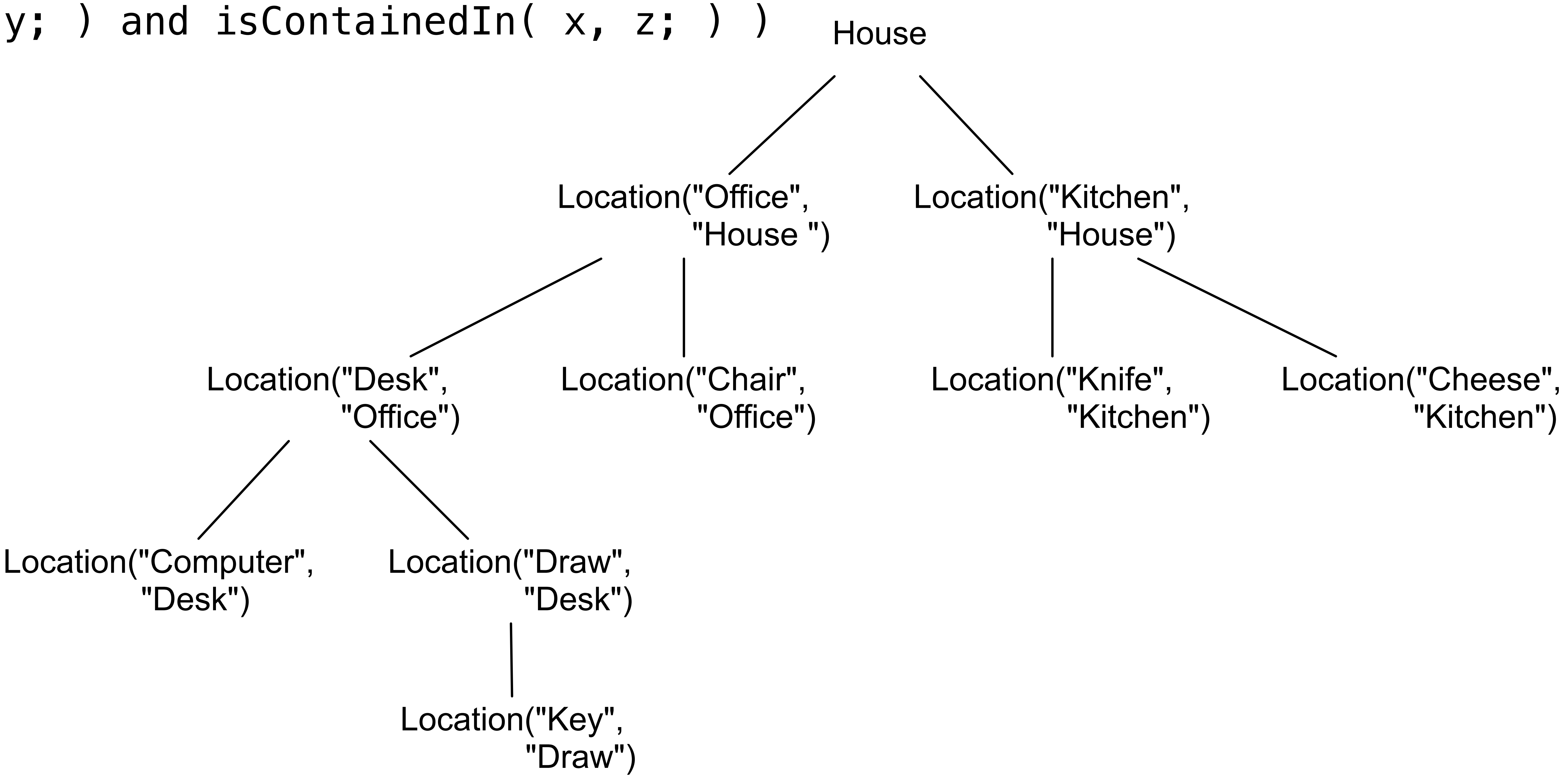

House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

Backward Chaining

```
rule "go" salience 10
when
    $s : String(  )
then
    System.out.println( $s );
end
```

```
rule "go1"
when
    String( this == "go1" )
    isContainedIn("Office", "House"; )
then
    System.out.println( "office is in the house" );
end
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```


House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

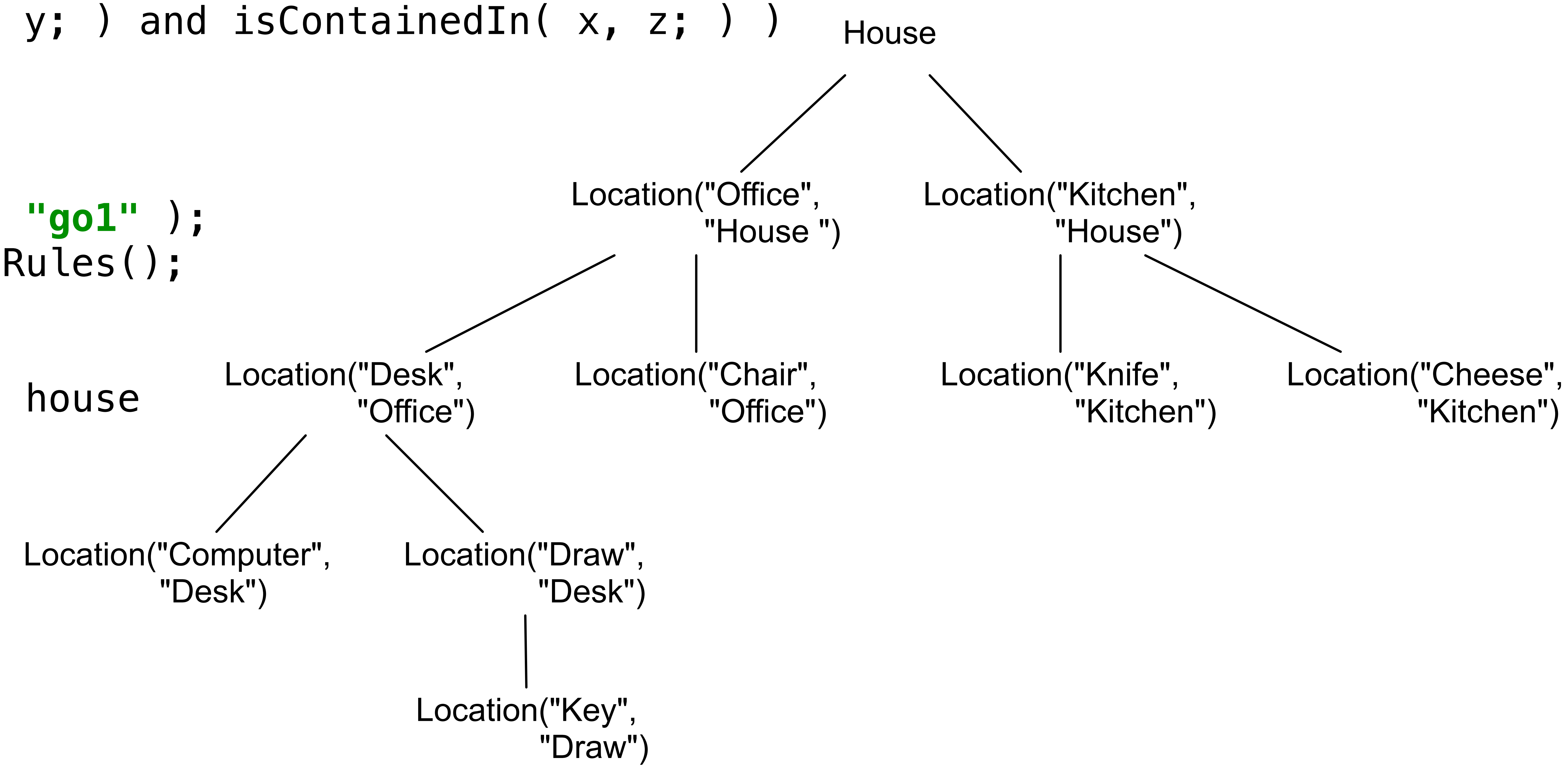
Backward Chaining

```
rule "go" salience 10
when
    $s : String(  )
then
    System.out.println( $s );
end
```

```
rule "go1"
when
    String( this == "go1" )
    isContainedIn("Office", "House"; )
then
    System.out.println( "office is in the house" );
end
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

```
ksession.insert( "go1" );
ksession.fireAllRules();
---
go1
office is in the house
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

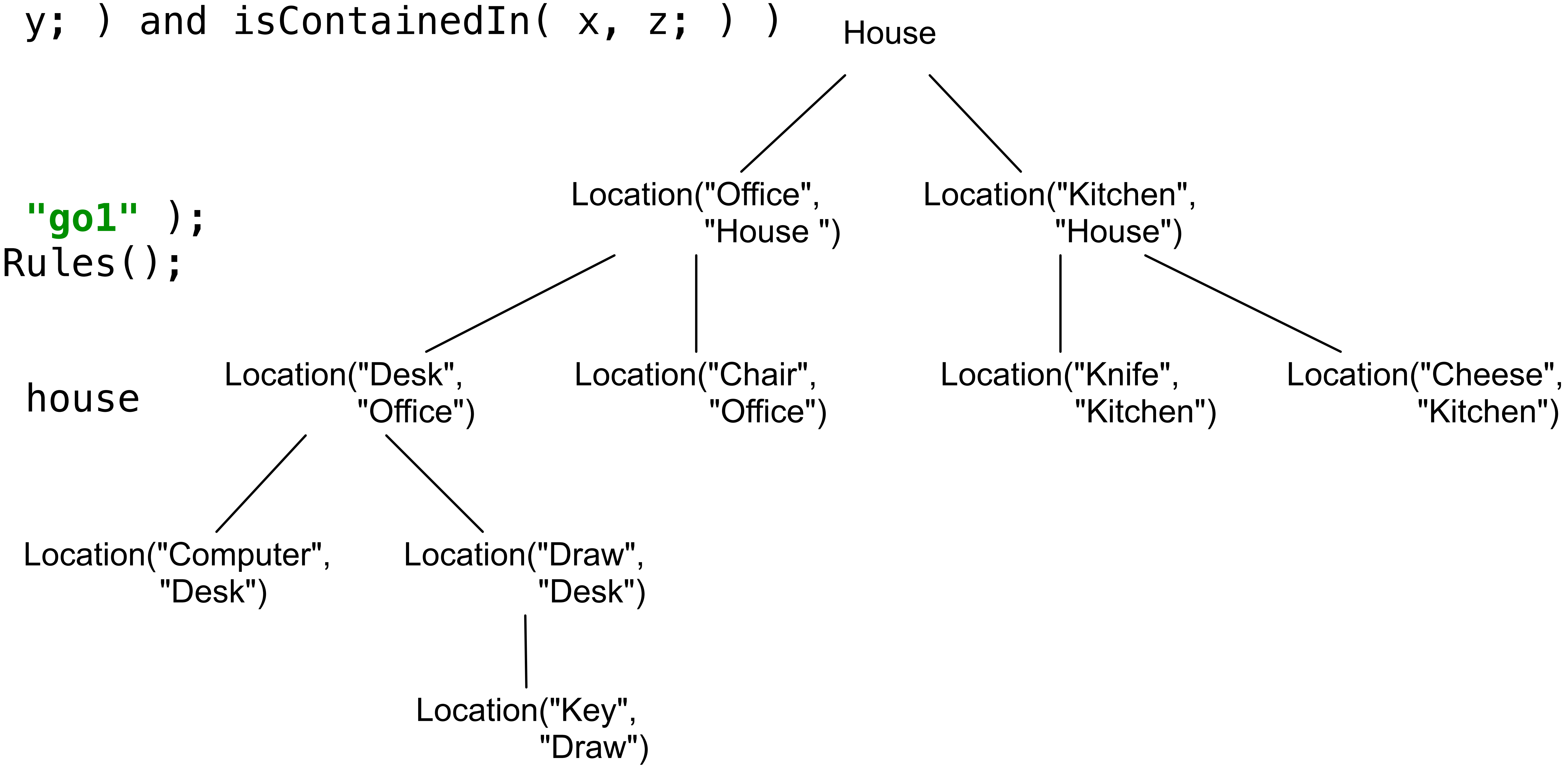
```
rule "go" salience 10
when
    $s : String(  )
then
    System.out.println( $s );
end
```

```
rule "go1"
when
    String( this == "go1" )
    isContainedIn("Office", "House"; )
then
    System.out.println( "office is in the house" );
end
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

isContainedIn(x==Office, y==House)

```
ksession.insert( "go1" );
ksession.fireAllRules();
---
go1
office is in the house
```


House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

Backward Chaining

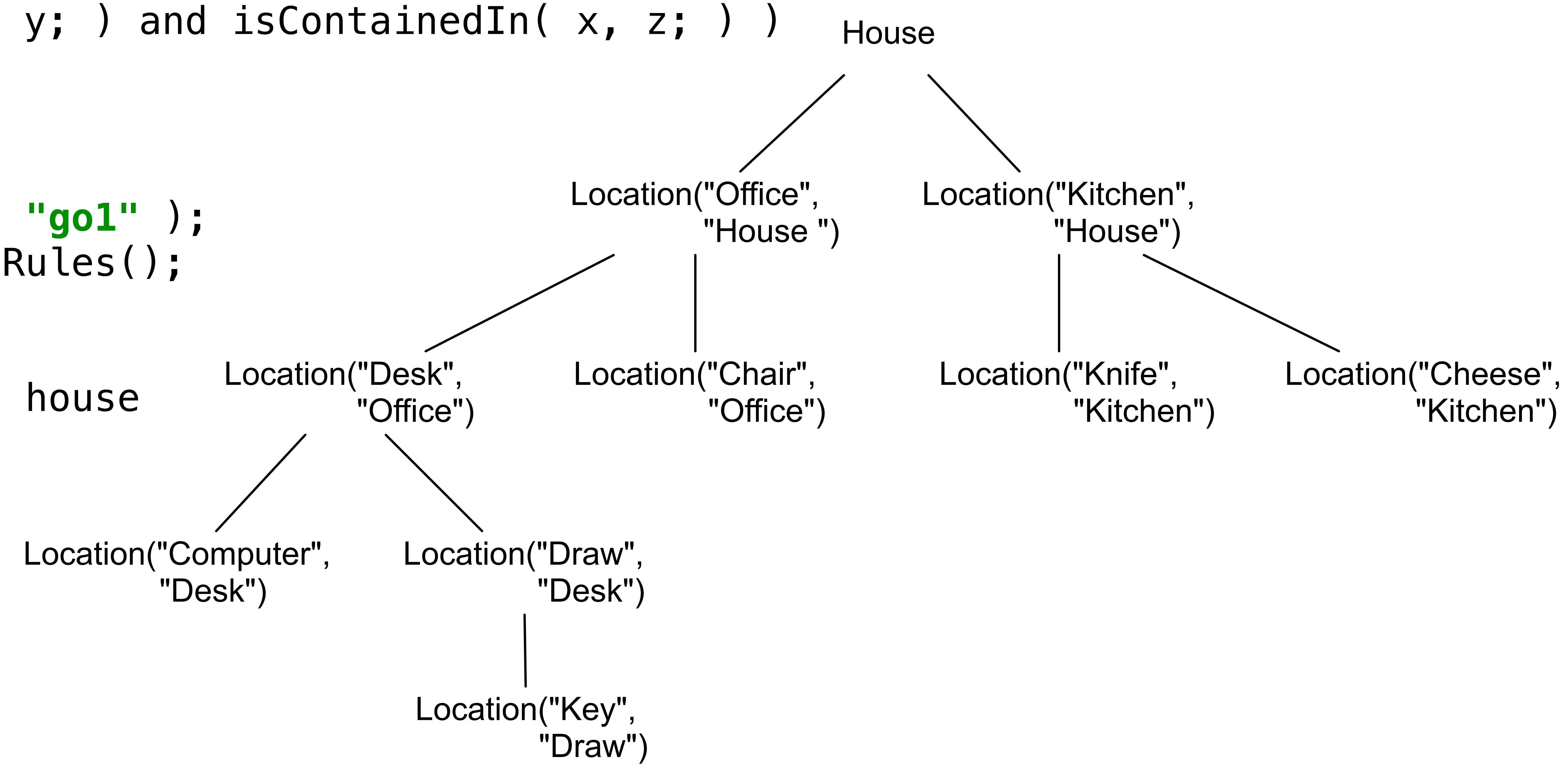
```
rule "go" salience 10
when
    $s : String(  )
then
    System.out.println( $s );
end
```

```
rule "go1"
when
    String( this == "go1" )
    isContainedIn("Office", "House"; )
then
    System.out.println( "office is in the house" );
end
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

```
isContainedIn(x==Office, y==House)
Location(x==Office, y==House)
```

```
ksession.insert( "go1" );
ksession.fireAllRules();
---
go1
office is in the house
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

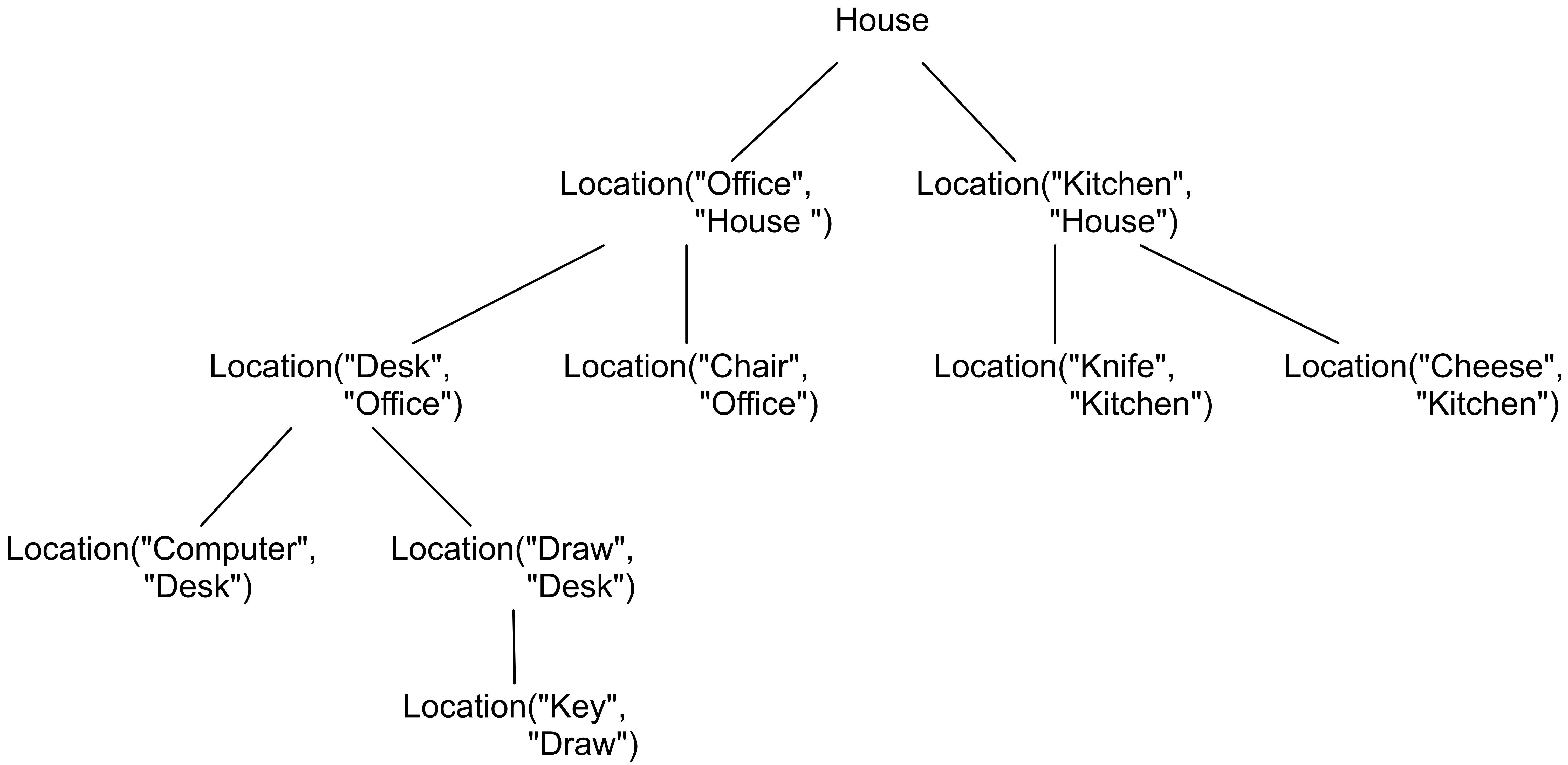
Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

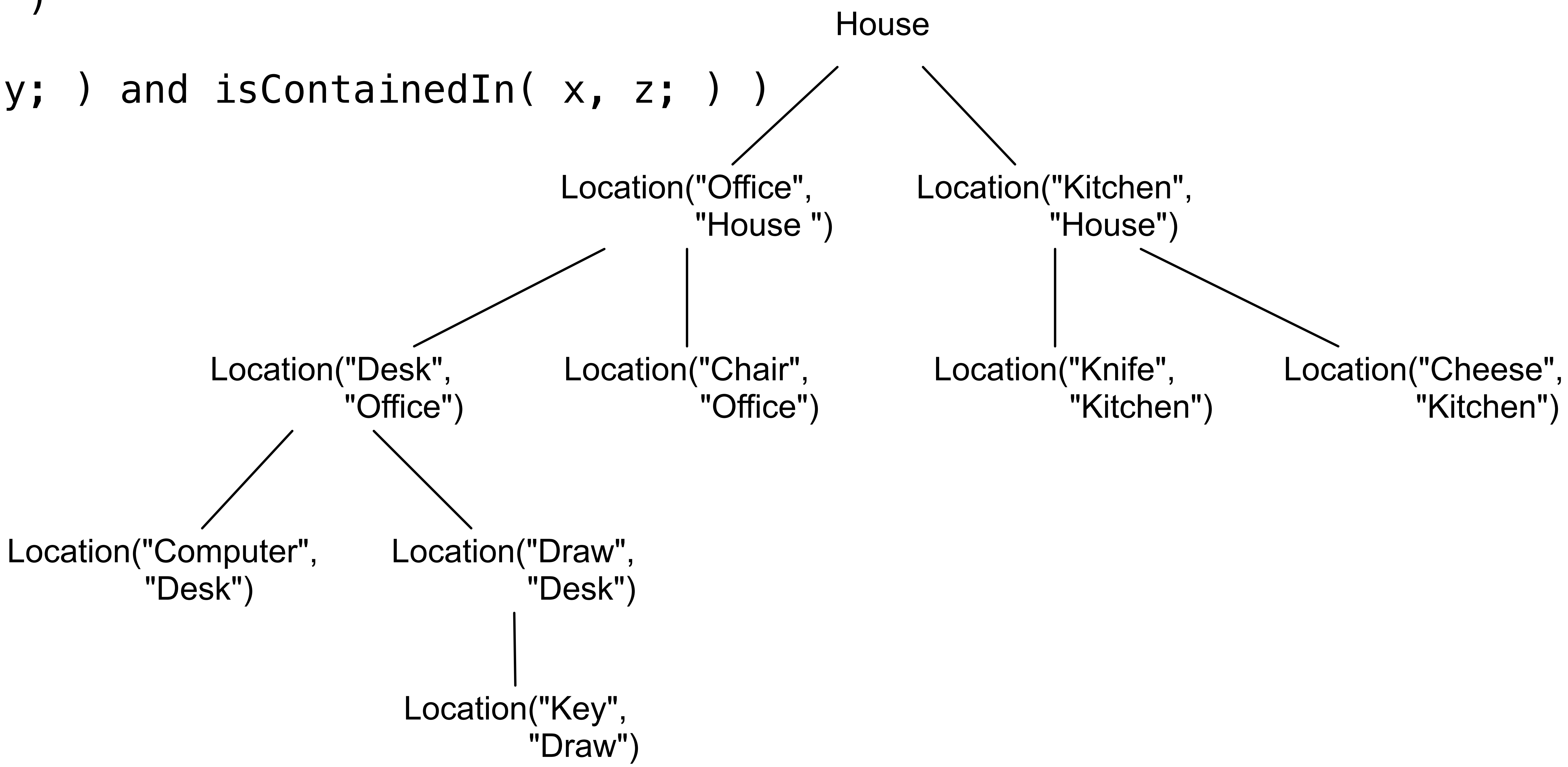
Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

Location("Draw", "Desk")

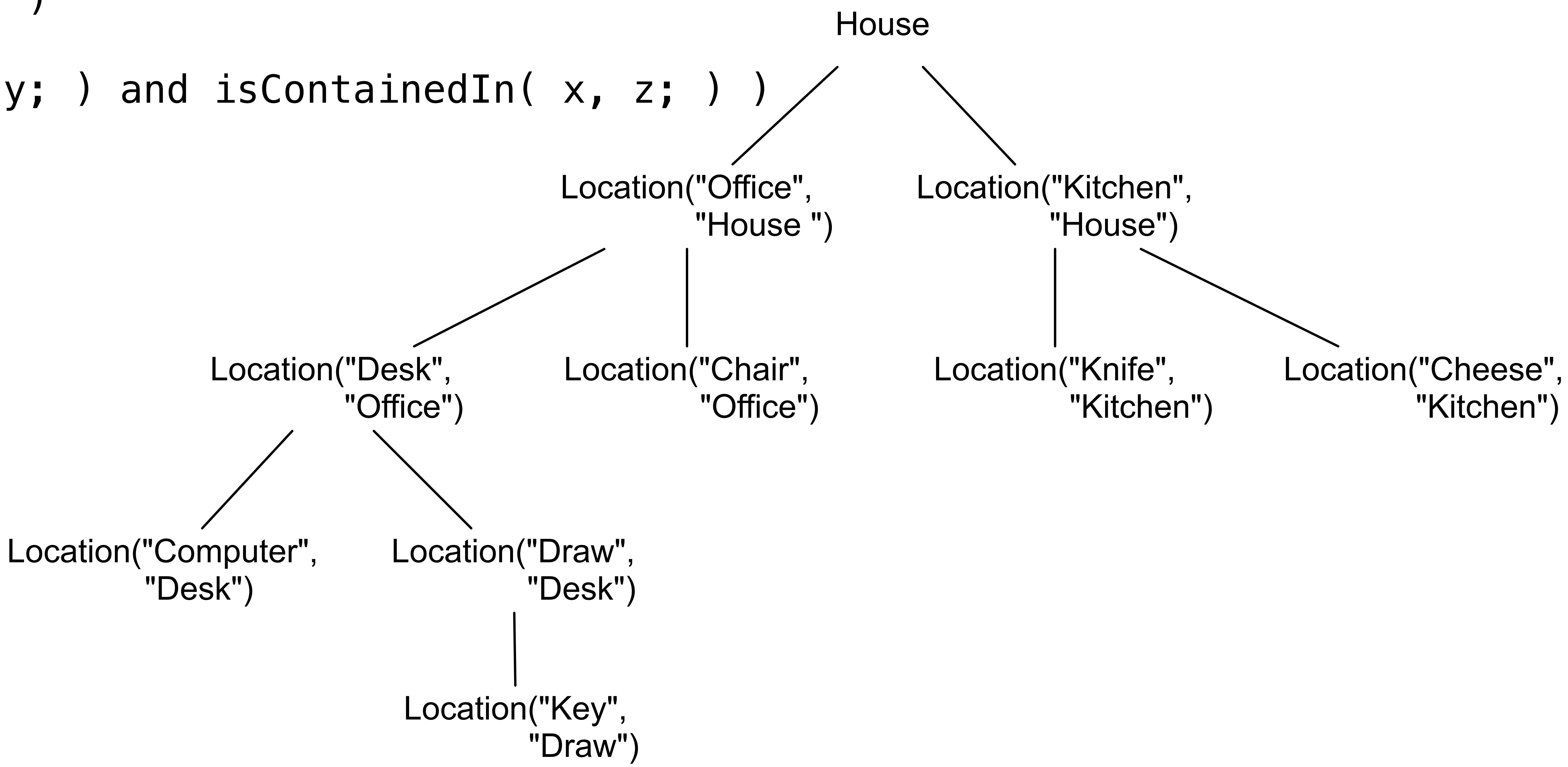
Location("Key", "Draw")

Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```



House

Location("Office",
"House ")

Location("Kitchen",
"House")

Location("Desk",
"Office")

Location("Chair",
"Office")

Location("Knife",
"Kitchen")

Location("Cheese",
"Kitchen")

Location("Computer",
"Desk")

Location("Draw",
"Desk")

Location("Key",
"Draw")

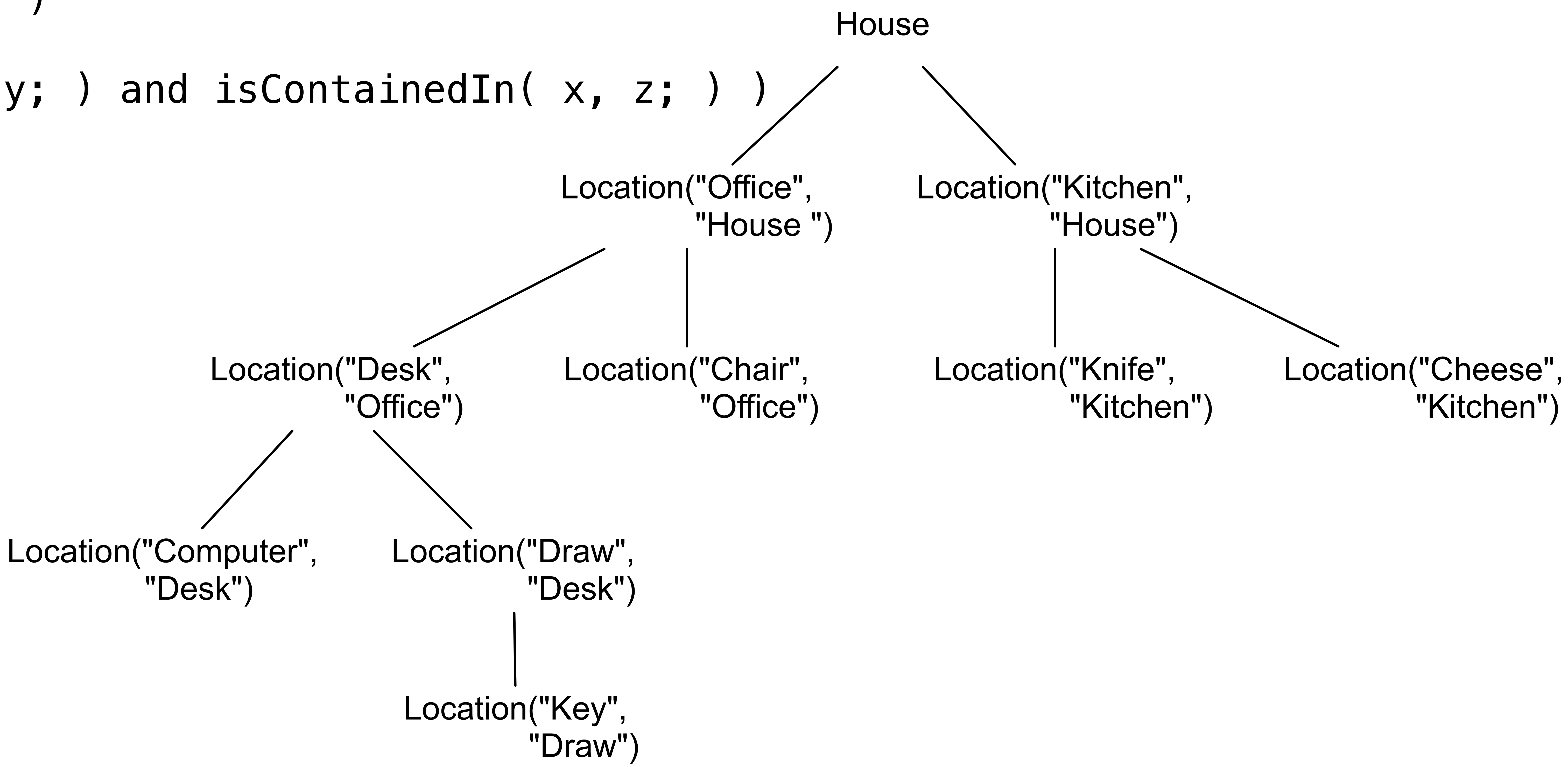
Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
 isContainedIn(x==Draw, y==House)
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

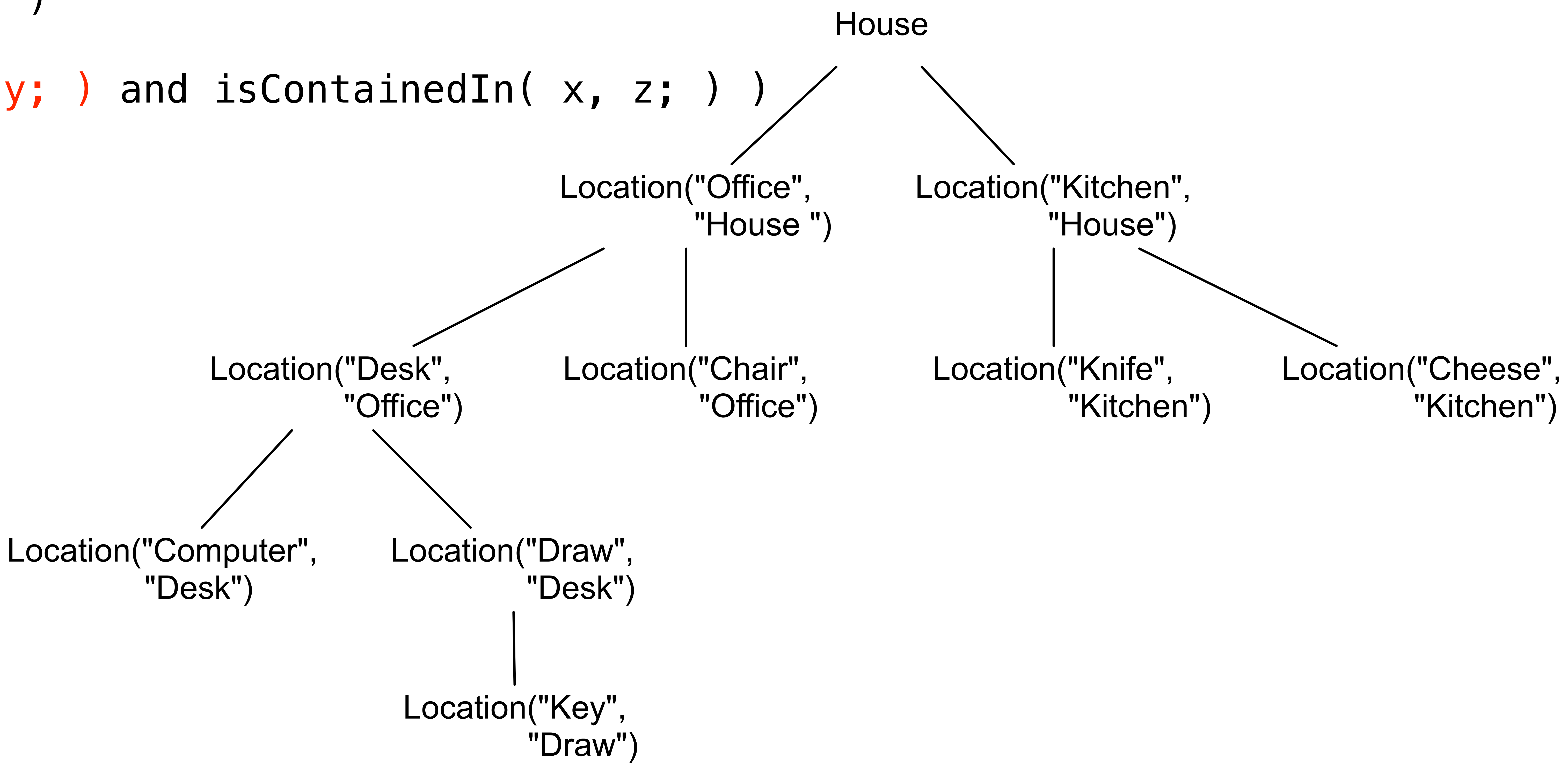
```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
isContainedIn(x==Draw, y==House)
```

```
Location(z==Office, y==House)
```


House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

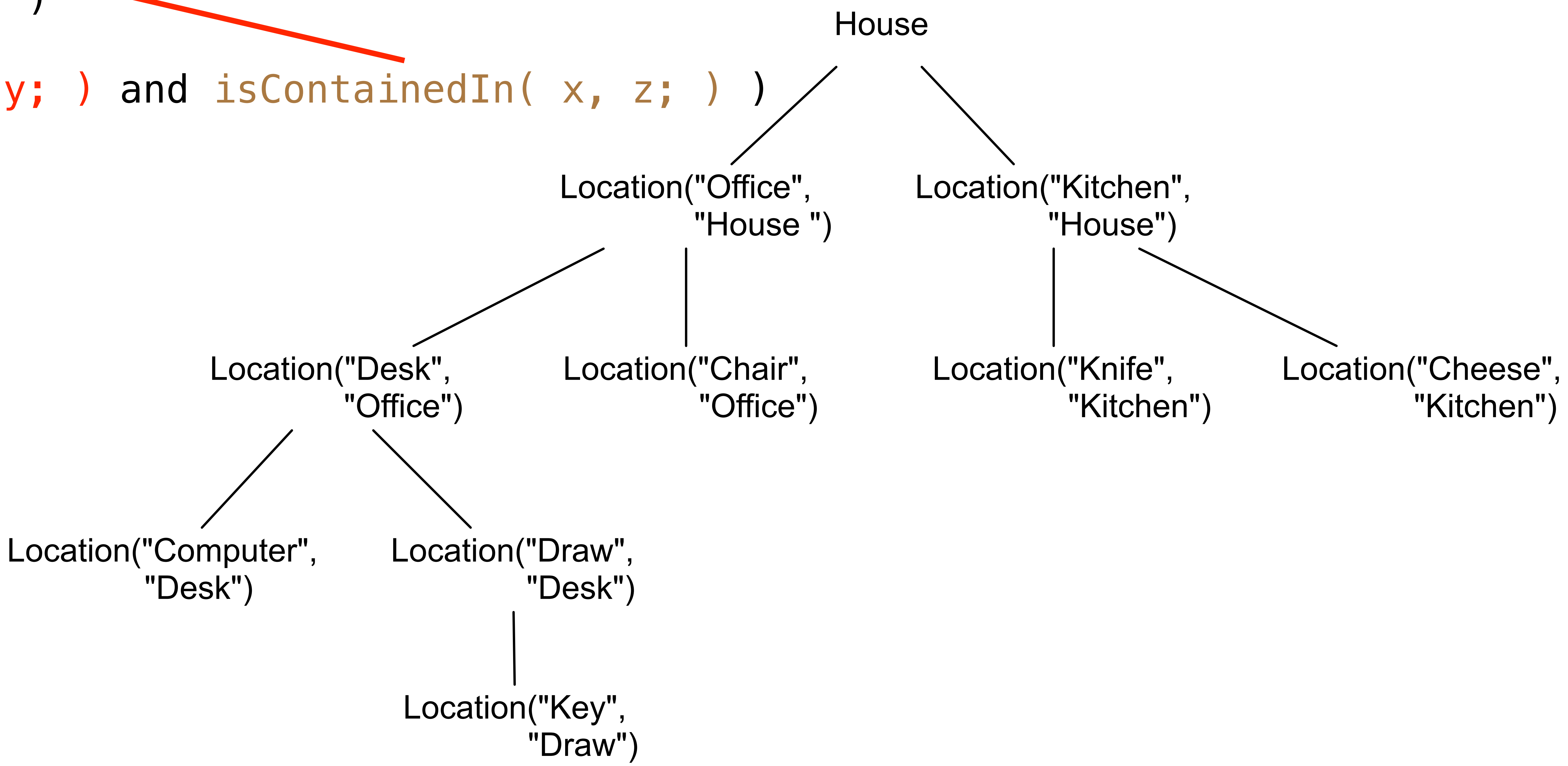
Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

```
isContainedIn(x==Draw, y==House)
Location(z==Office, y==House)
isContainedIn(x==Draw, z==Office)
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

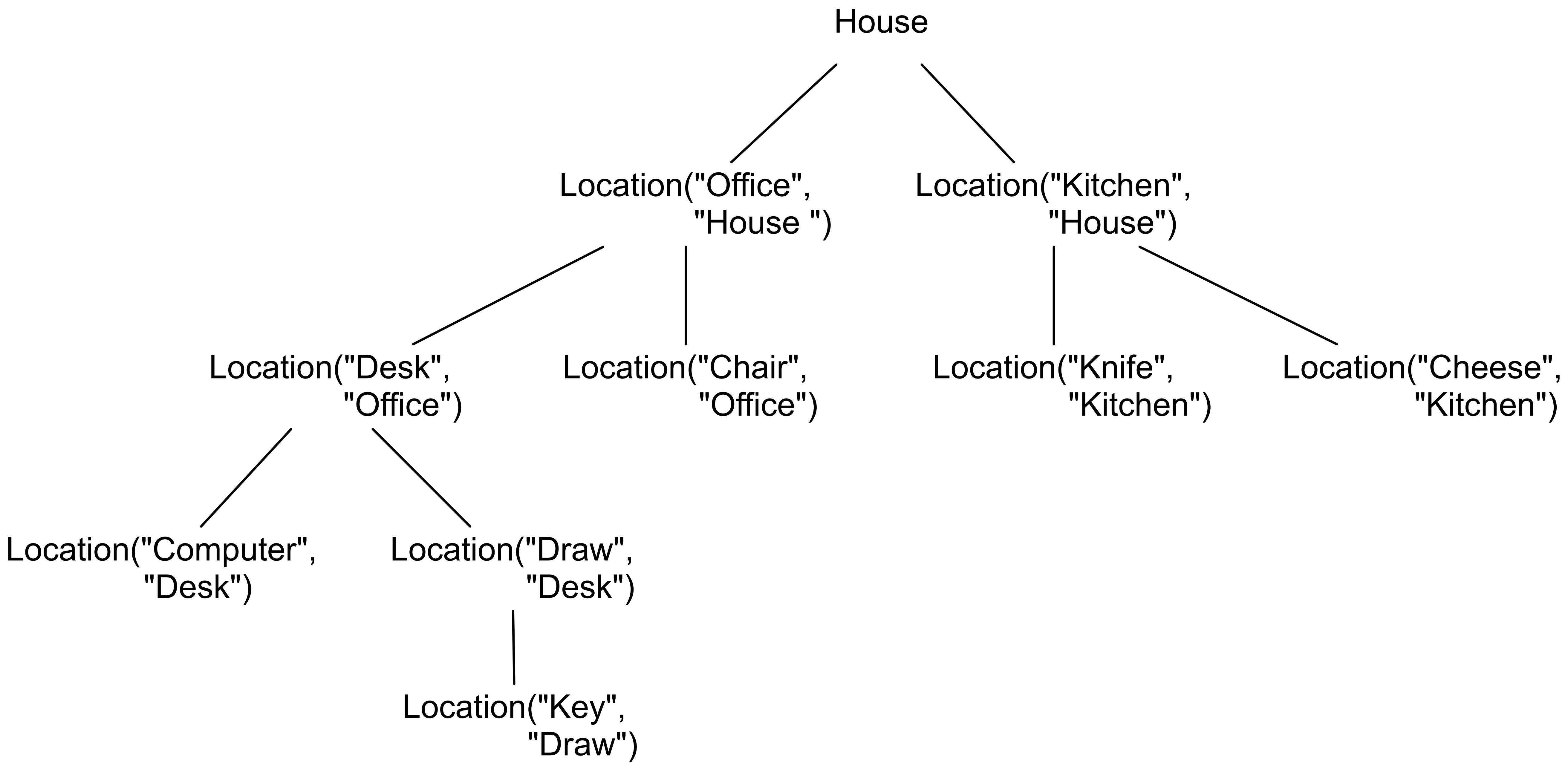
Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
 isContainedIn(x==Draw, y==House)
 Location(z==Office, y==House)
 isContainedIn(x==Draw, z==Office)
   Location(z==Kitchen, y==House)
 isContainedIn(x==Draw, z==Kitchen)
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

Location("Draw", "Desk")

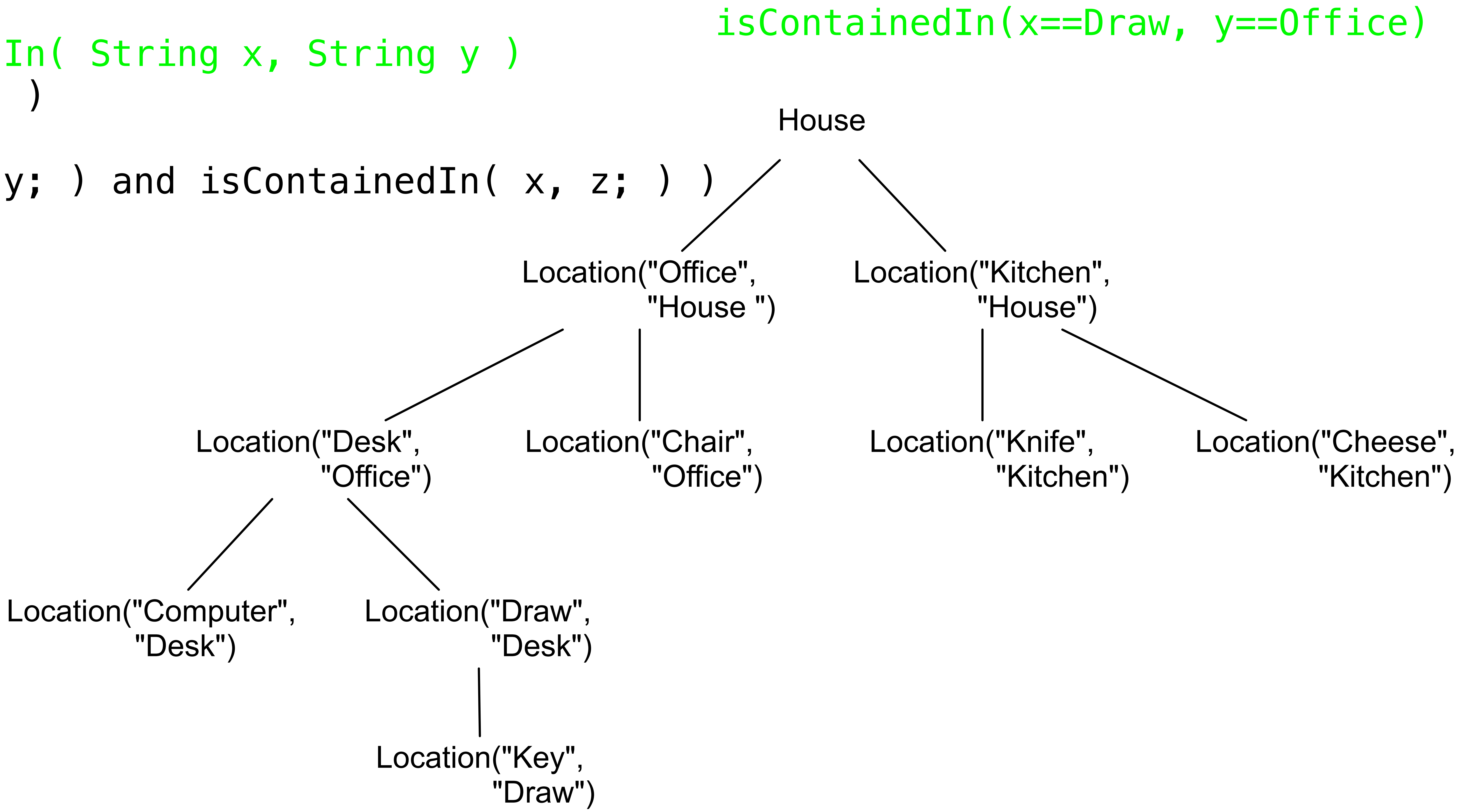
Location("Key", "Draw")

Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```


isContainedIn(x==Draw, y==Office)
House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

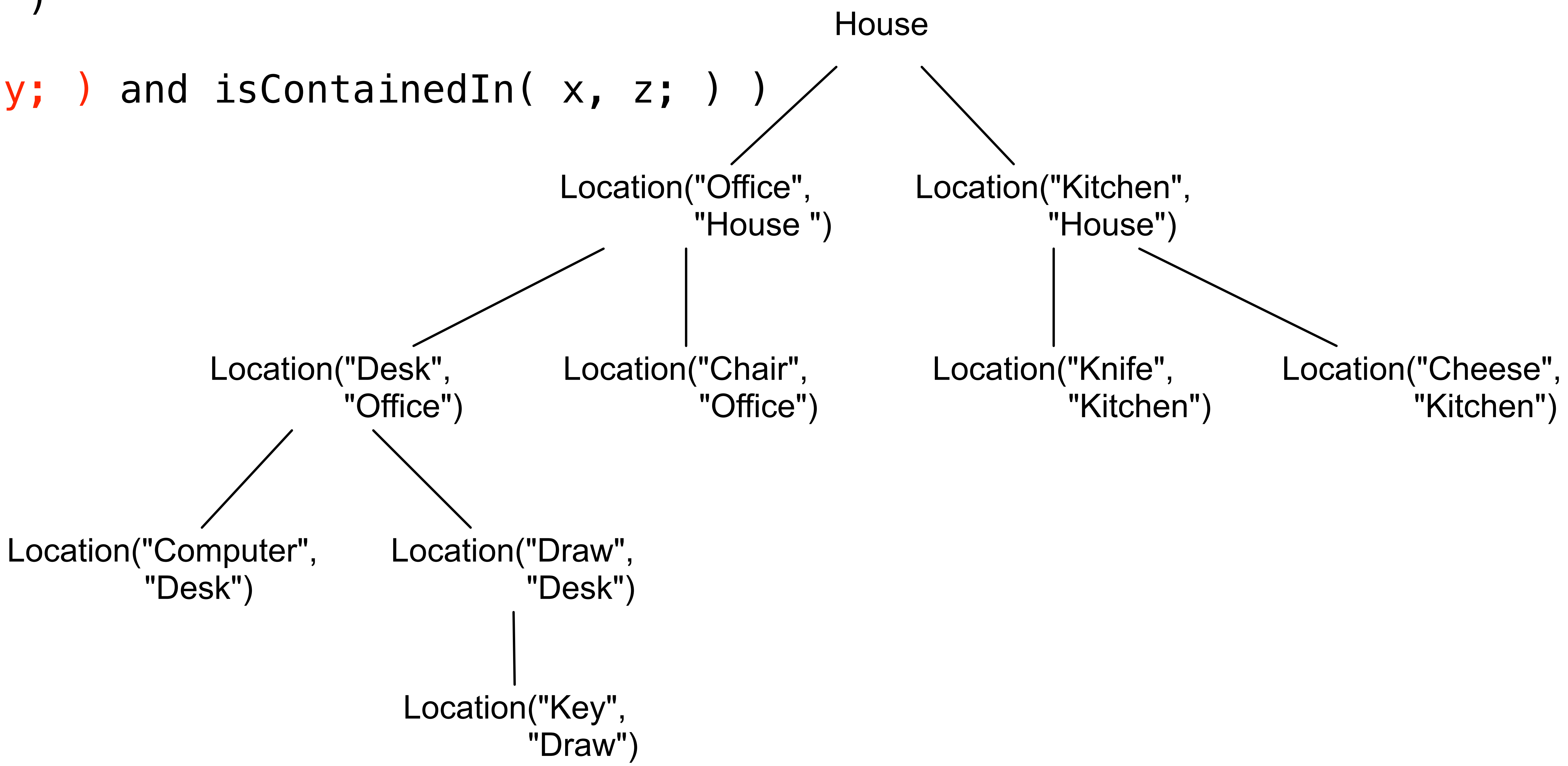
Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

```
isContainedIn(x==Draw, y==Office)
Location(z==Desk, y==Office)
```


House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

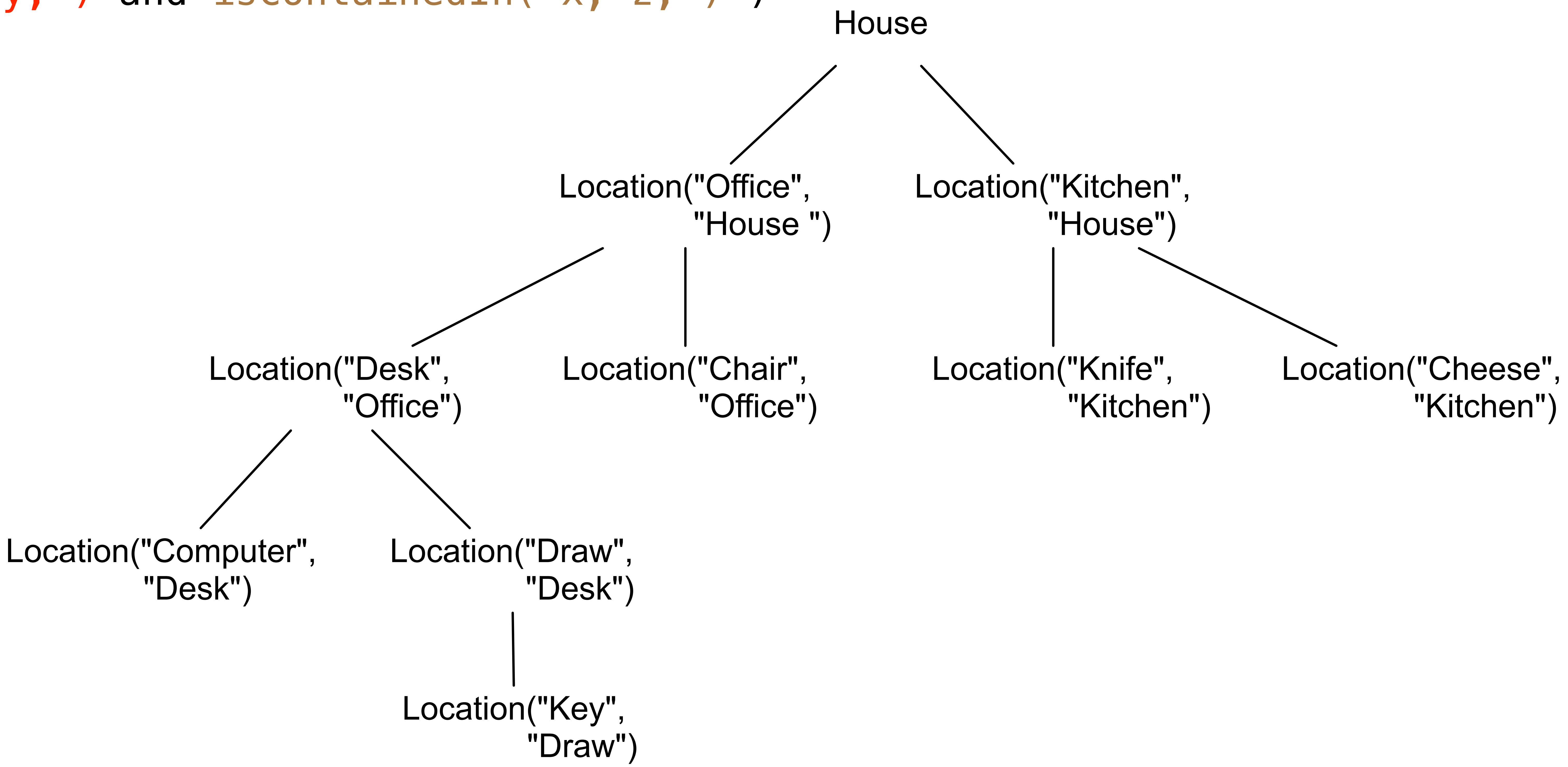
Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

```
isContainedIn(x==Draw, y==Office)
Location(z==Desk, y==Office)
isContainedIn(x==Draw, z==Desk)
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

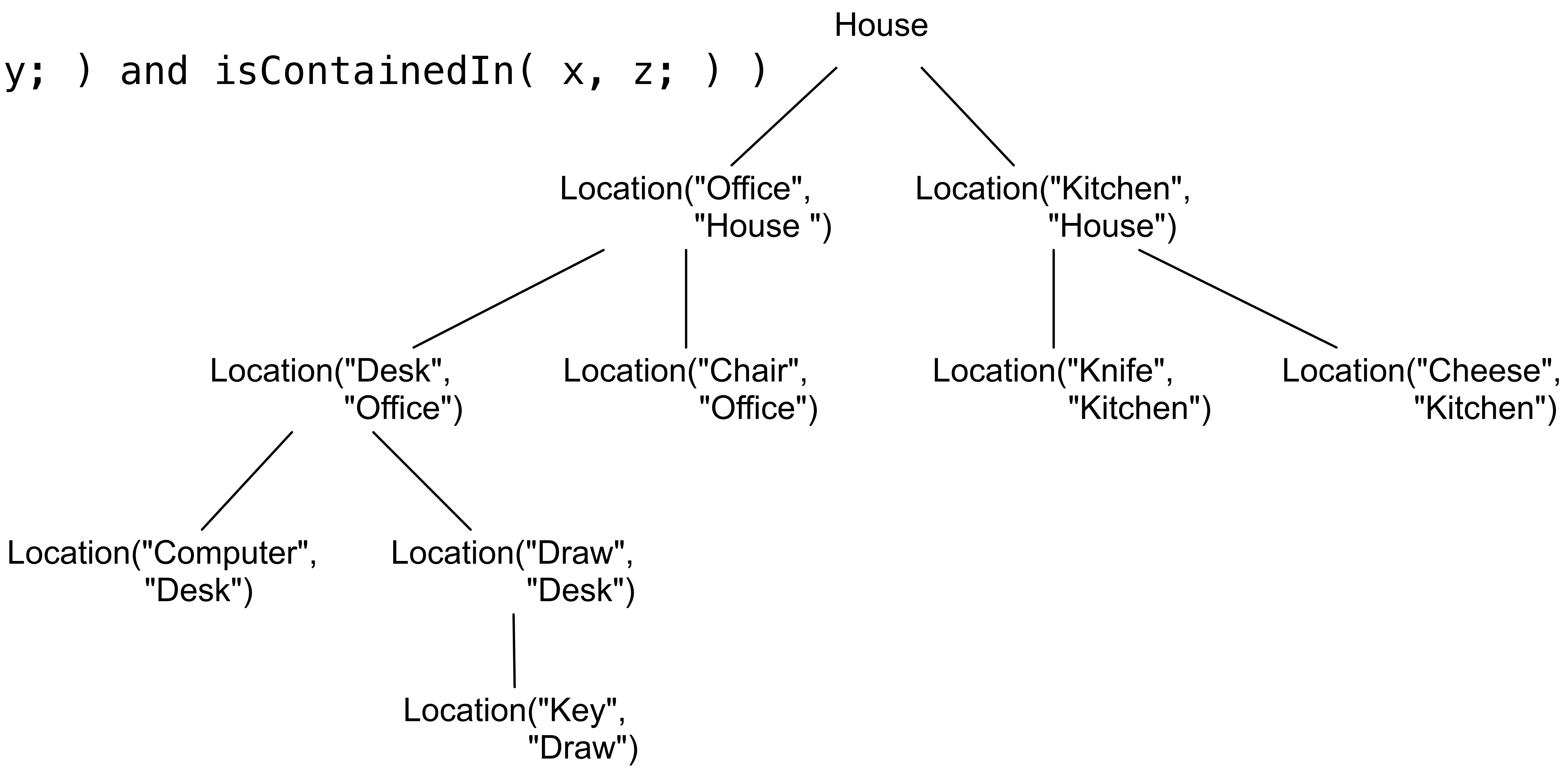
Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

isContainedIn(x==Draw, y==Desk)



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

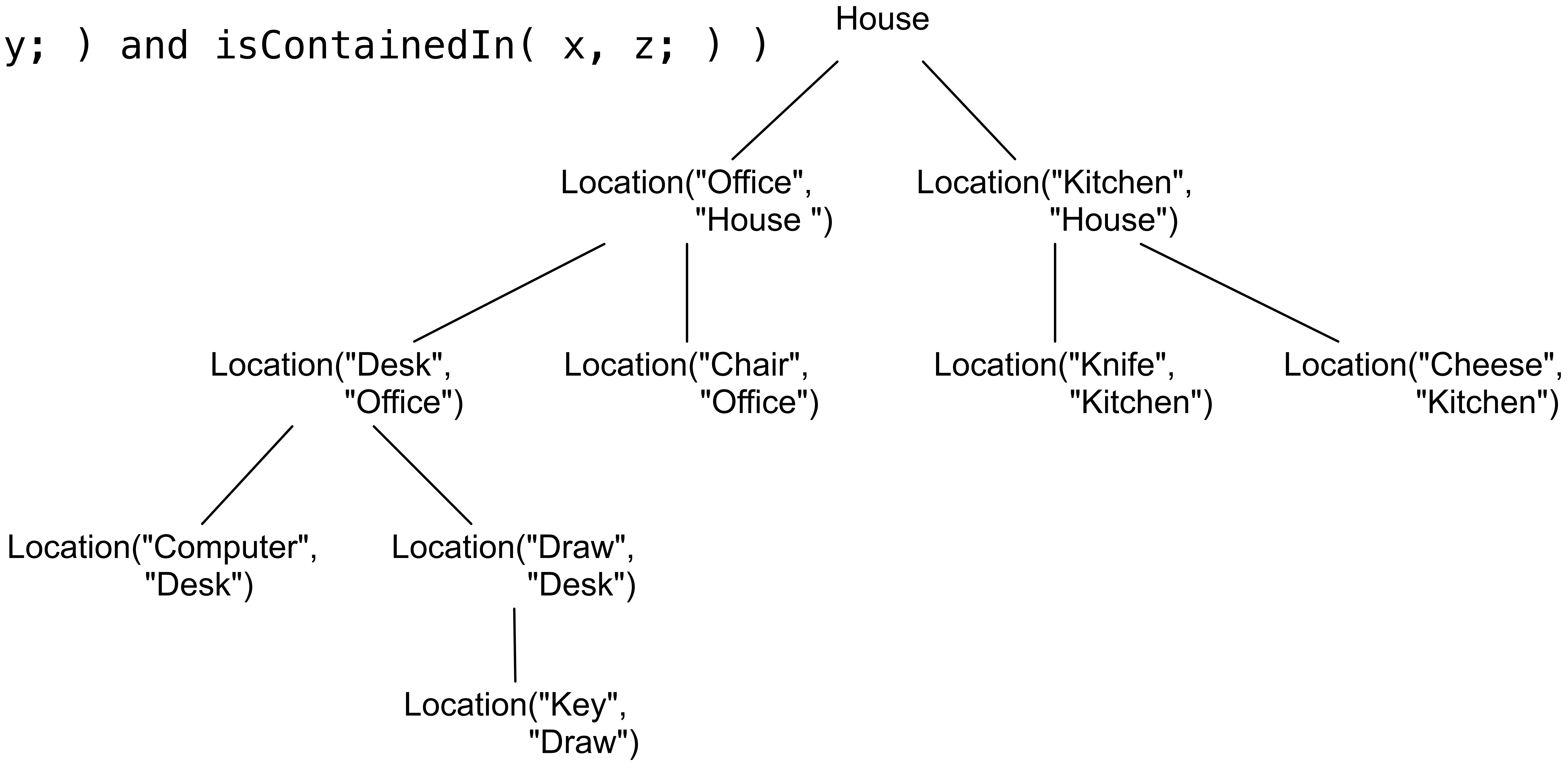
Backward Chaining

```
rule "go2"
when
    String( this == "go2" )
    isContainedIn("Draw", "House"; )
then
    System.out.println( "Draw in the House" );
end
```

```
ksession.insert( "go2" );
ksession.fireAllRules();
---
go2
Draw in the House
```

```
query isContainedIn( String x, String y )
  Location( x, y; )
  or
  ( Location( z, y; ) and isContainedIn( x, z; ) )
end
```

```
isContainedIn(x==Draw, y==Desk)
Location(x==Draw, y==Desk)
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

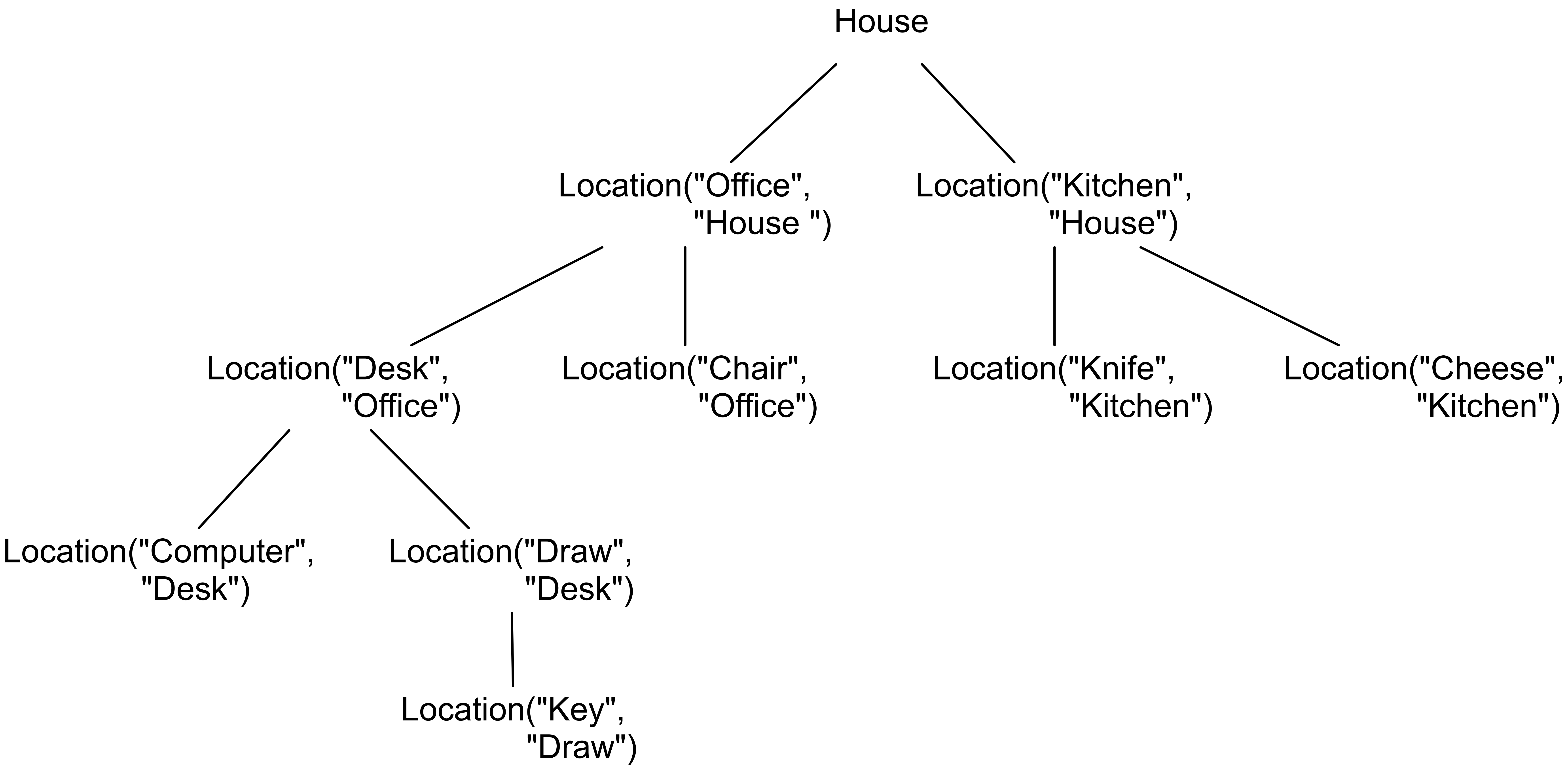
Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

```
rule "go3"
when
    String( this == "go3" )
    isContainedIn("Key", "Office"; )
then
    System.out.println( "Key in the Office" );
end
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

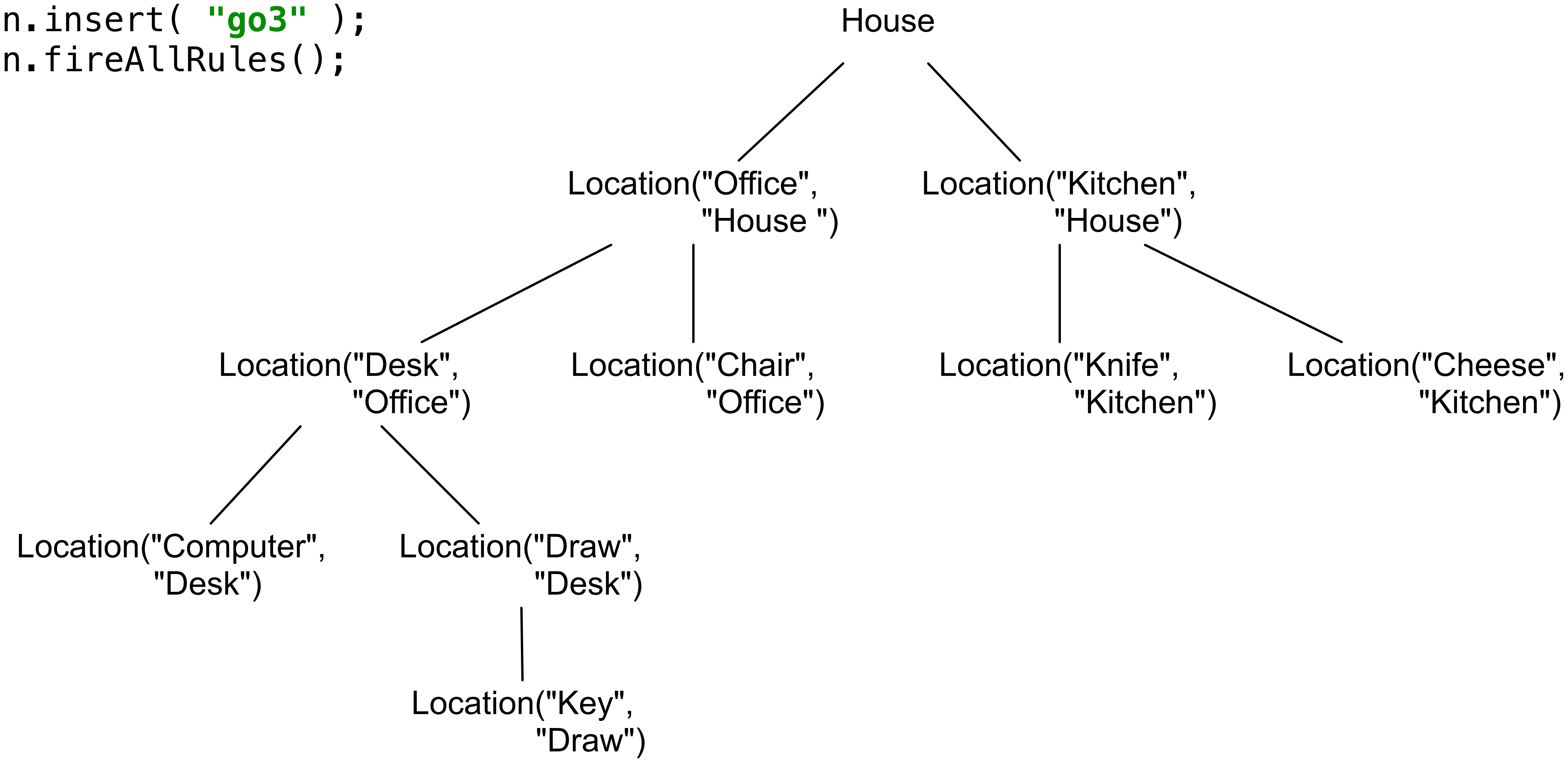
Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

```
rule "go3"
when
    String( this == "go3" )
    isContainedIn("Key", "Office"; )
then
    System.out.println( "Key in the Office" );
end
```

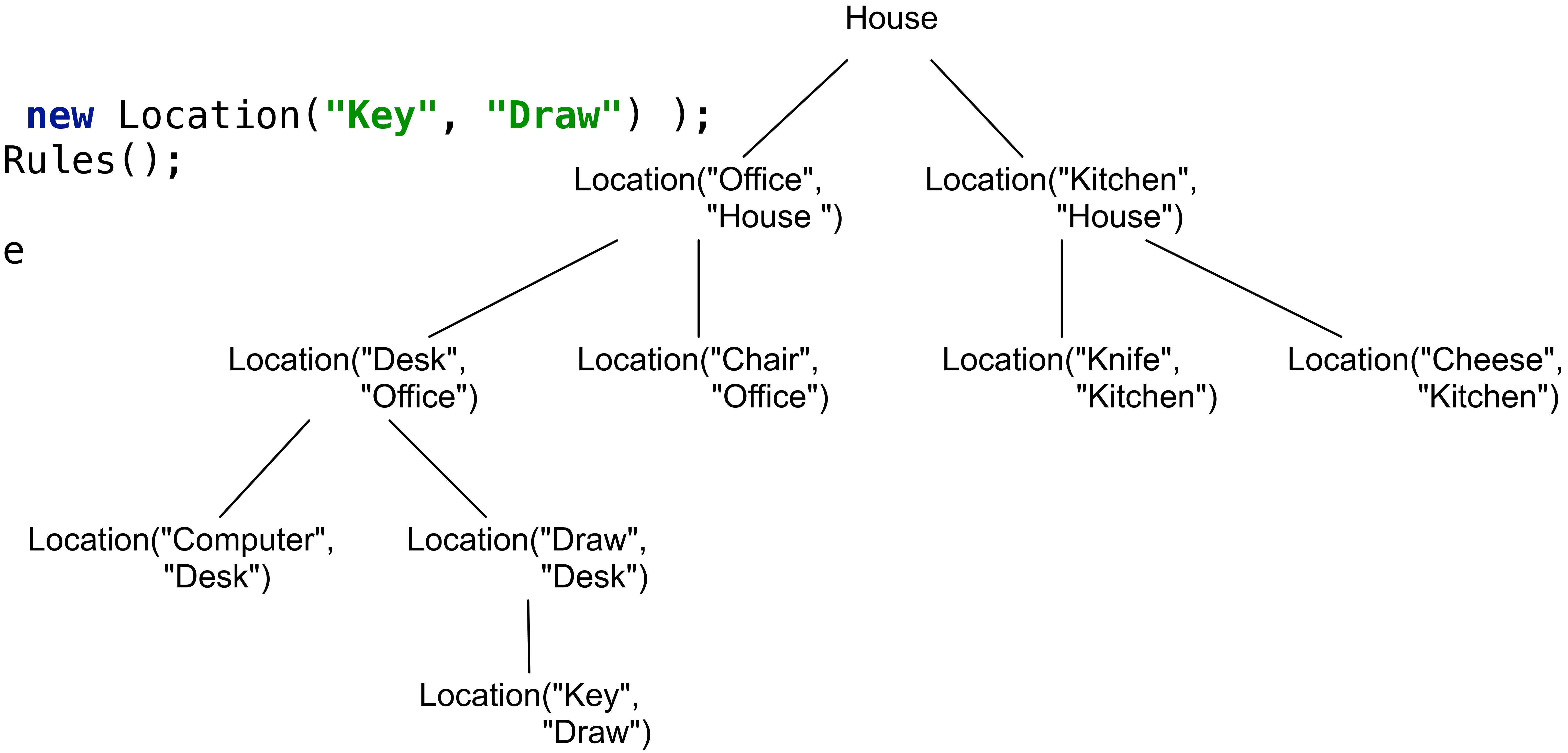
```
ksession.insert( "go3" );
ksession.fireAllRules();
---
go3
```



House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

Backward Chaining

```
rule "go3"
when
    String( this == "go3" )
    isContainedIn("Key", "Office"; )
then
    System.out.println( "Key in the Office" );
end

ksession.insert( "go3" );
ksession.fireAllRules();
---
go3


ksession.insert( new Location("Key", "Draw") );
ksession.fireAllRules();
---
Key in the Office
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

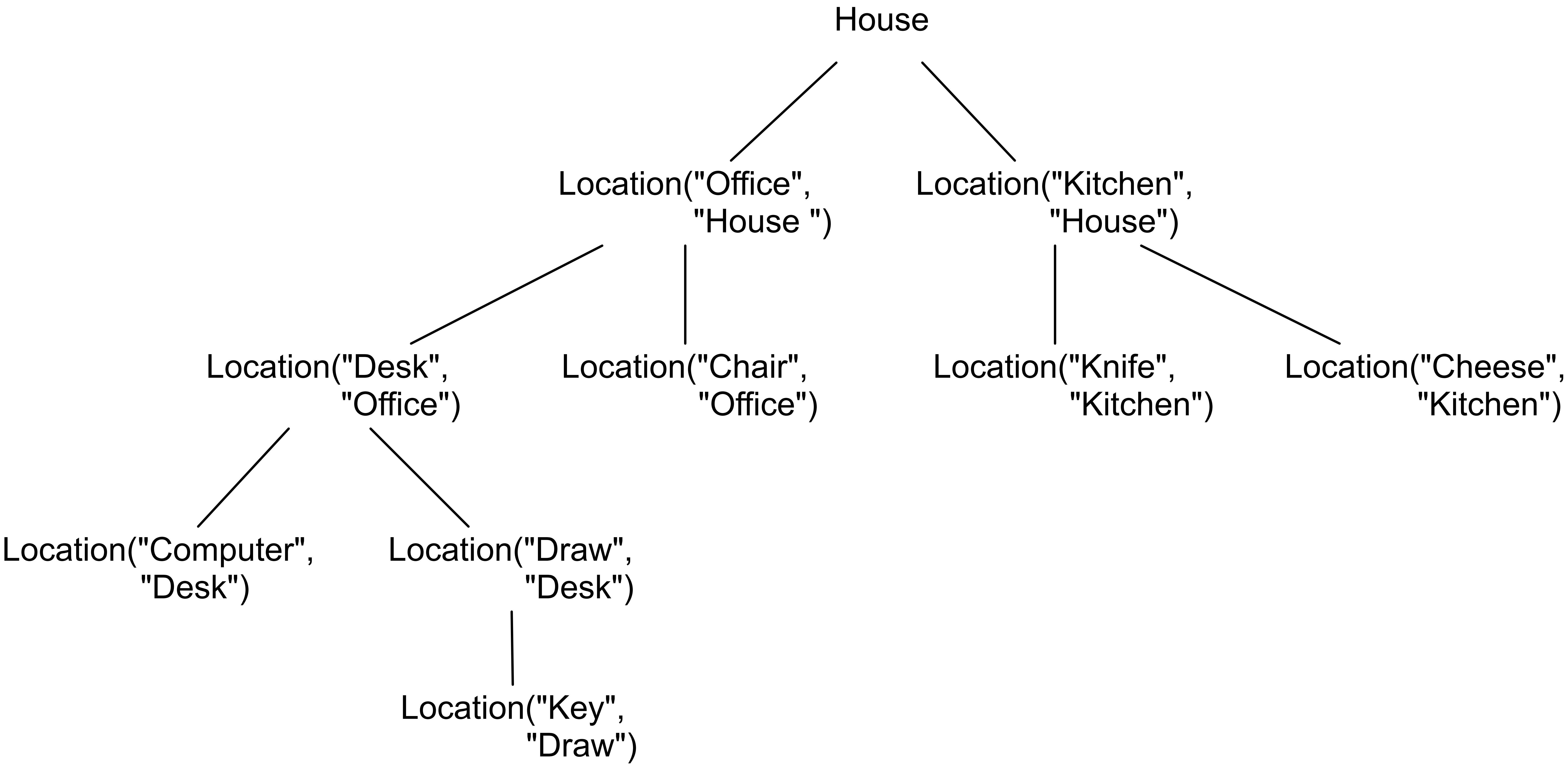
Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

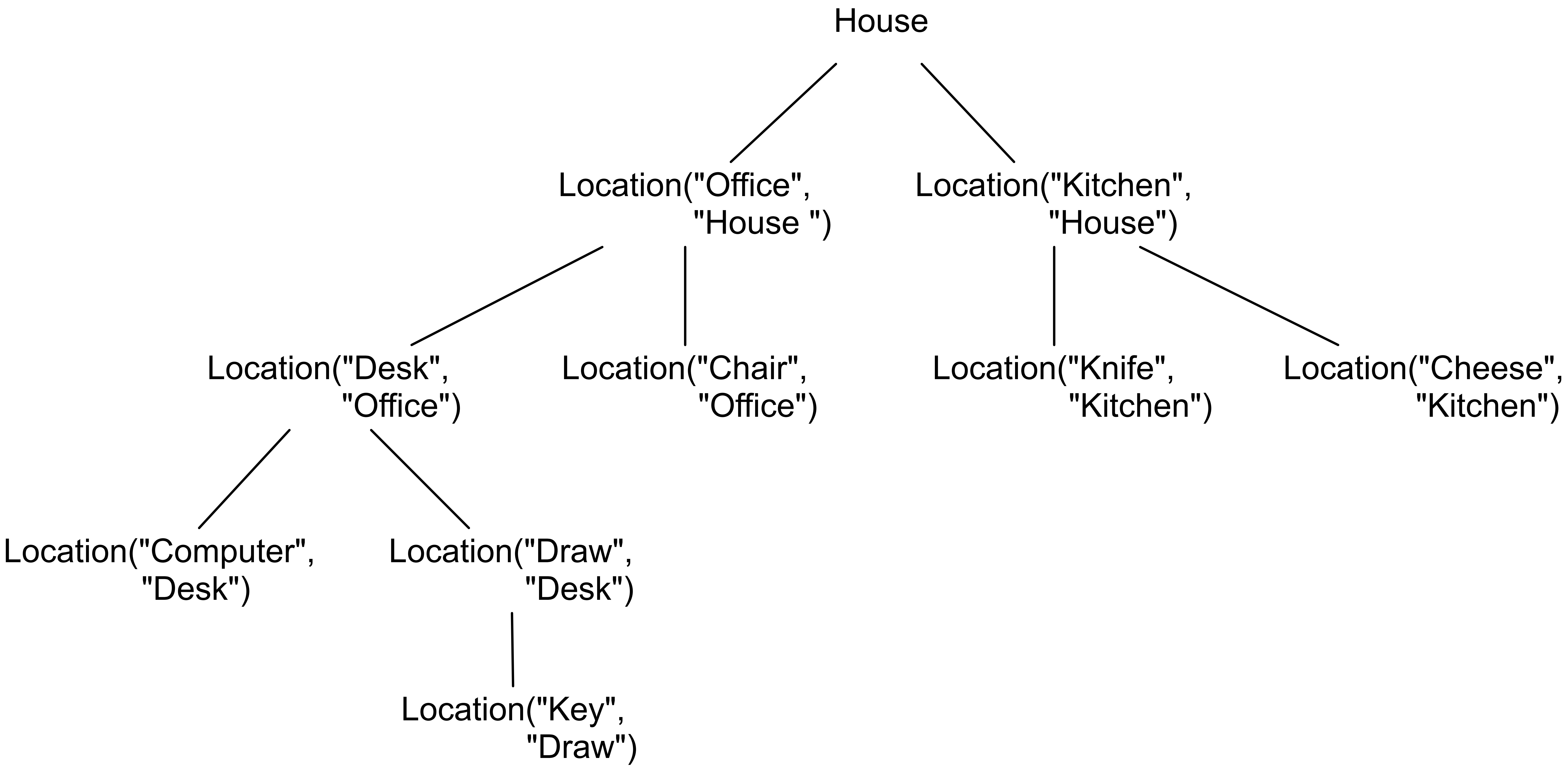
```
rule "go4"
when
    String( this == "go4" )
    isContainedIn(thing, "Office"; )
then
    System.out.println( "thing " + thing + " is in the Office" );
end
```


House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

Backward Chaining

```
rule "go4"
when
    String( this == "go4" )
    isContainedIn(thing, "Office"; )
then
    System.out.println( "thing " + thing + " is in the Office" );
end
```

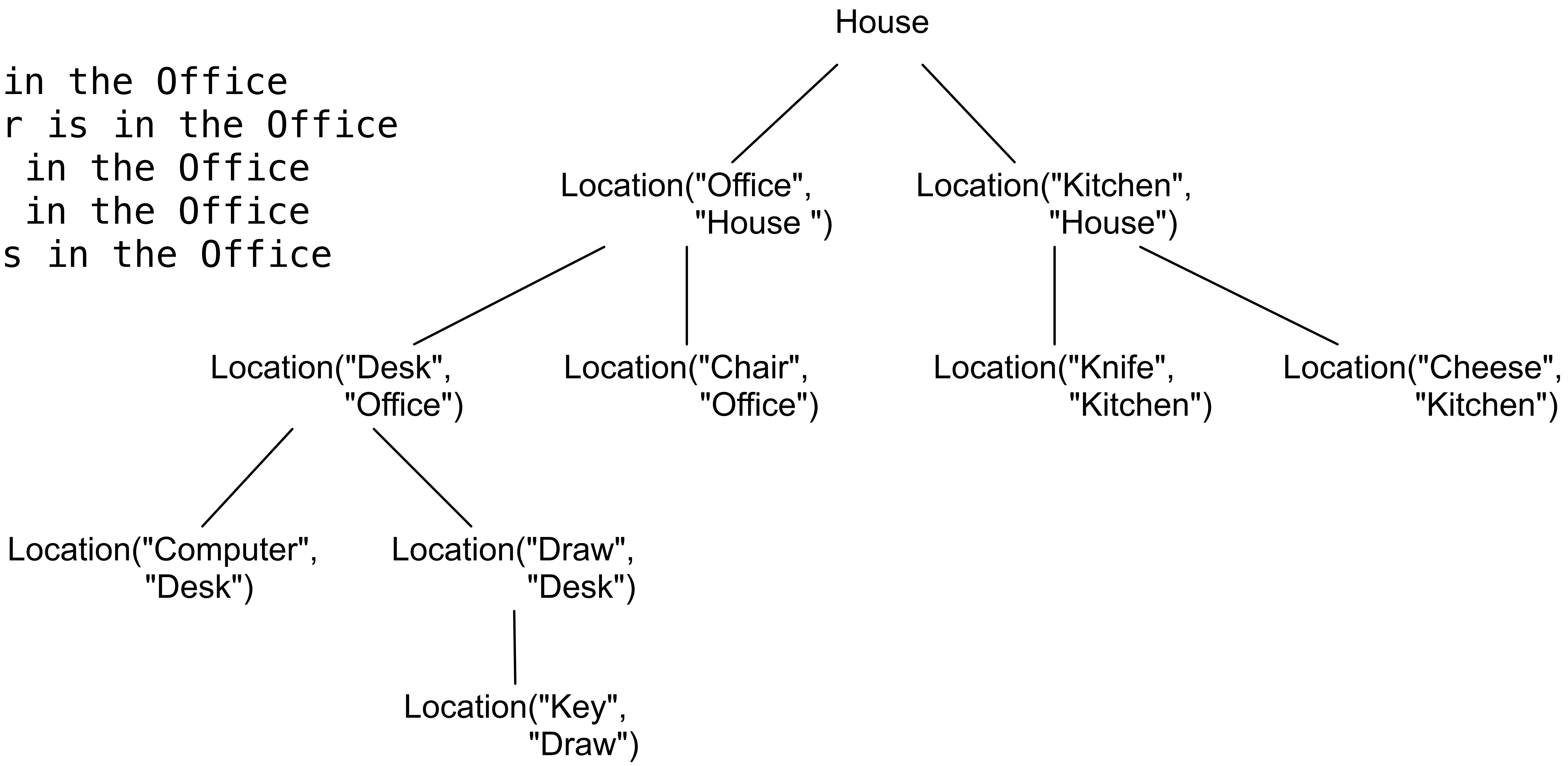
Out Var
(unbound)


House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

Backward Chaining

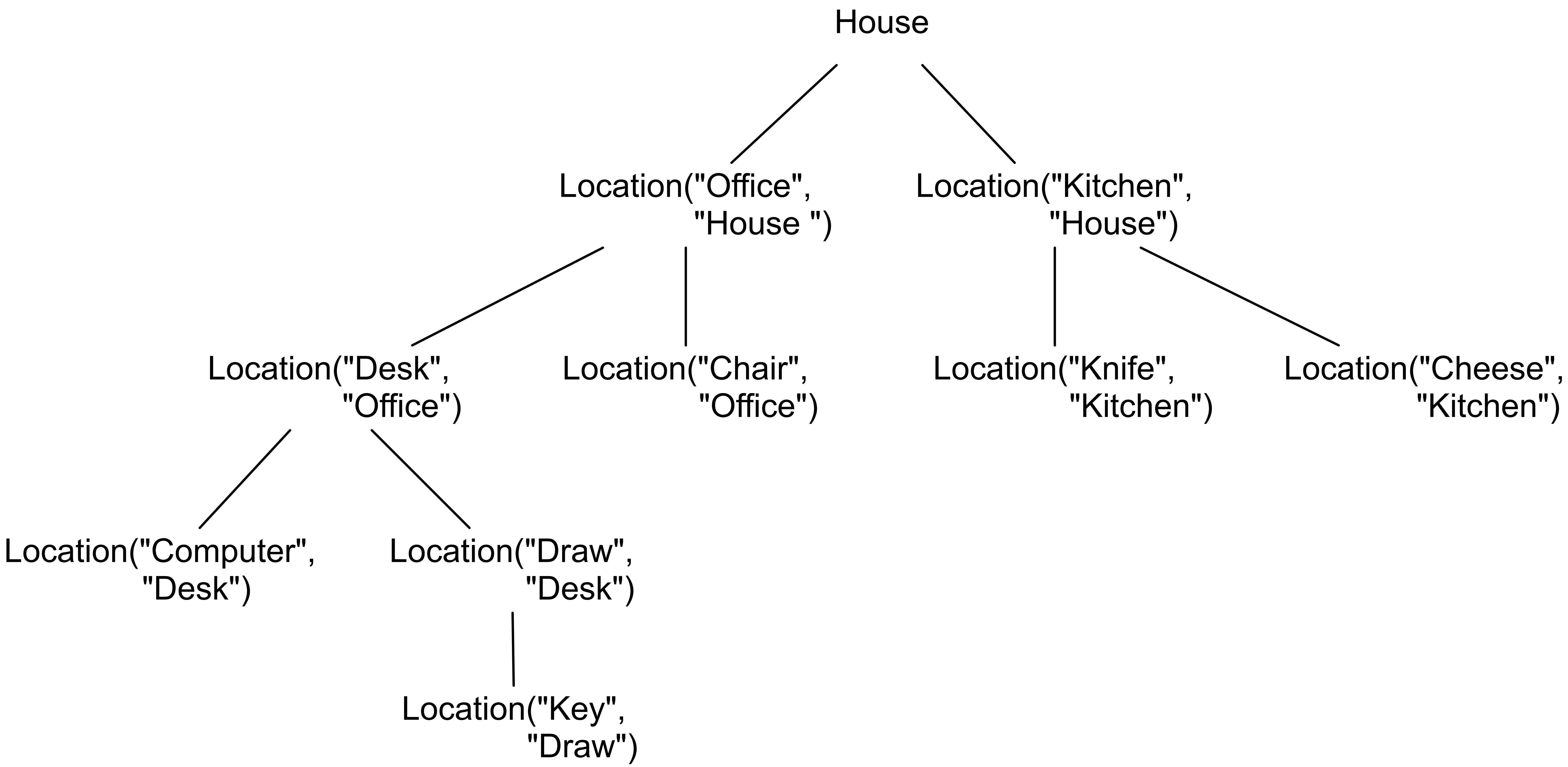
```
rule "go4"
when
    String( this == "go4" )
    isContainedIn(thing, "Office"; )
then
    System.out.println( "thing " + thing + " is in the Office" );
end

ksession.insert( "go4" );
ksession.fireAllRules();
---
go4
thing Key is in the Office
thing Computer is in the Office
thing Draw is in the Office
thing Desk is in the Office
thing Chair is in the Office
```

Out Var
(unbound)


House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

Backward Chaining

```
rule "go5"
when
    String( this == "go5" )
    isContainedIn(thing, location; )
then
    System.out.println( "thing " + thing + " is in " + location );
end
```



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

Location("Computer", "Desk")

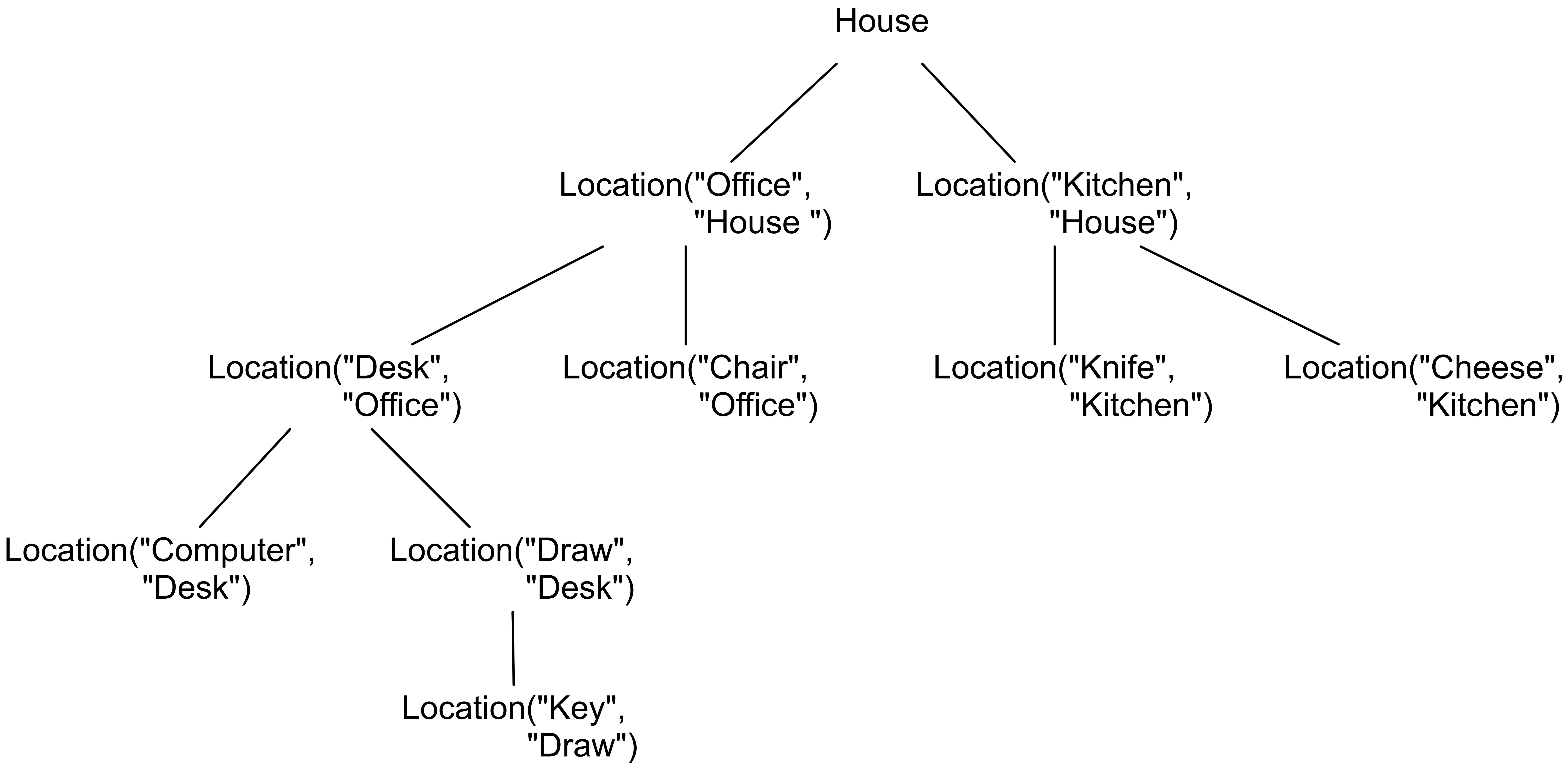
Location("Draw", "Desk")

Location("Key", "Draw")

Backward Chaining

```
rule "go5"
when
    String( this == "go5" )
    isContainedIn(thing, location; )
then
    System.out.println( "thing " + thing + " is in " + location );
end
```

Out Var
(unbound)



House

Location("Office",
"House ")

Location("Kitchen",
"House")

Location("Desk",
"Office")

Location("Chair",
"Office")

Location("Knife",
"Kitchen")

Location("Cheese",
"Kitchen")

Location("Computer",
"Desk")

Location("Draw",
"Desk")

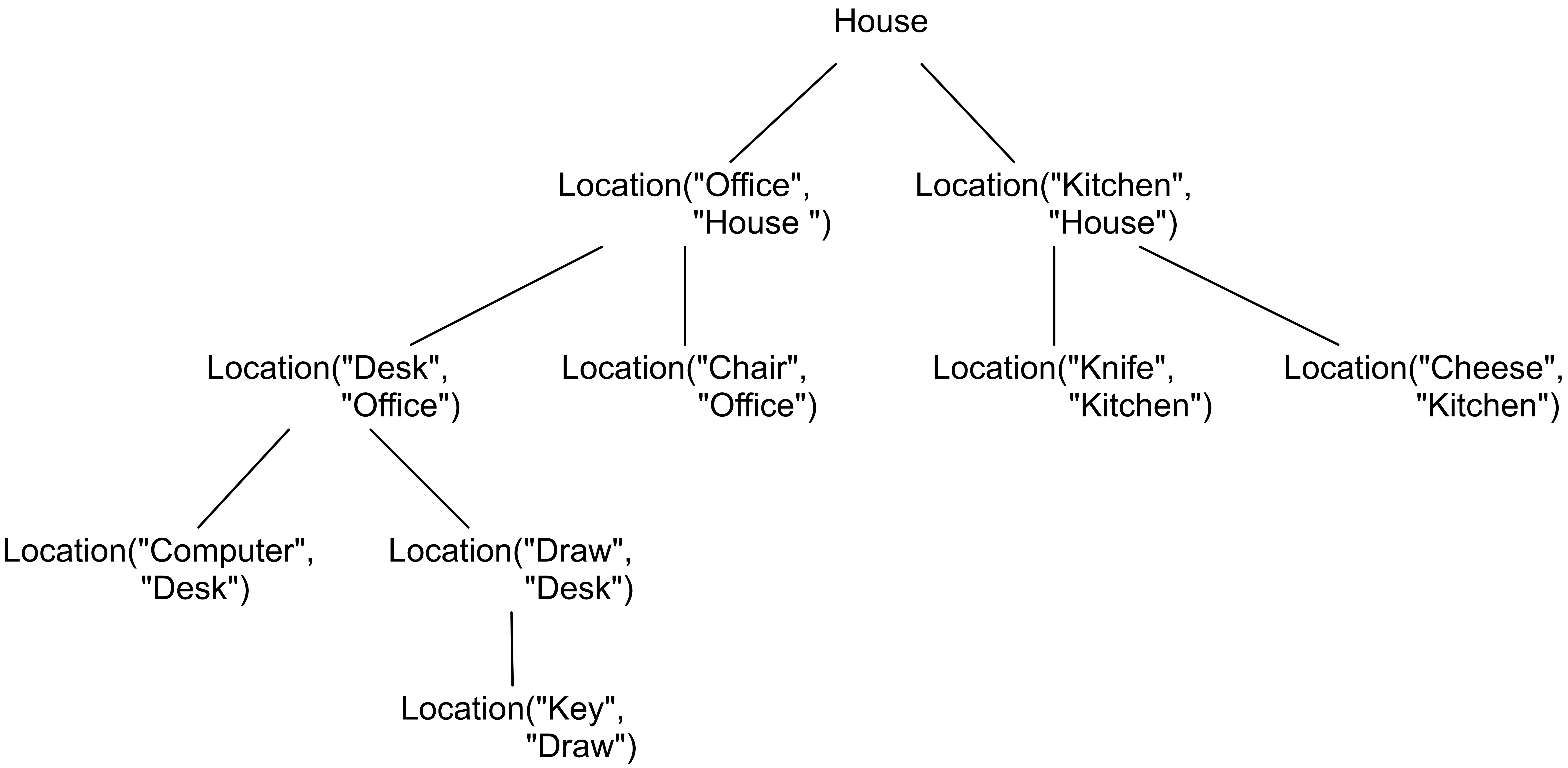
Location("Key",
"Draw")

Backward Chaining

```
rule "go5"
when
    String( this == "go5" )
    isContainedIn(thing, location; )
then
    System.out.println( "thing " + thing + " is in " + location );
end
```

Out Var (unbound)

Out Var (unbound)

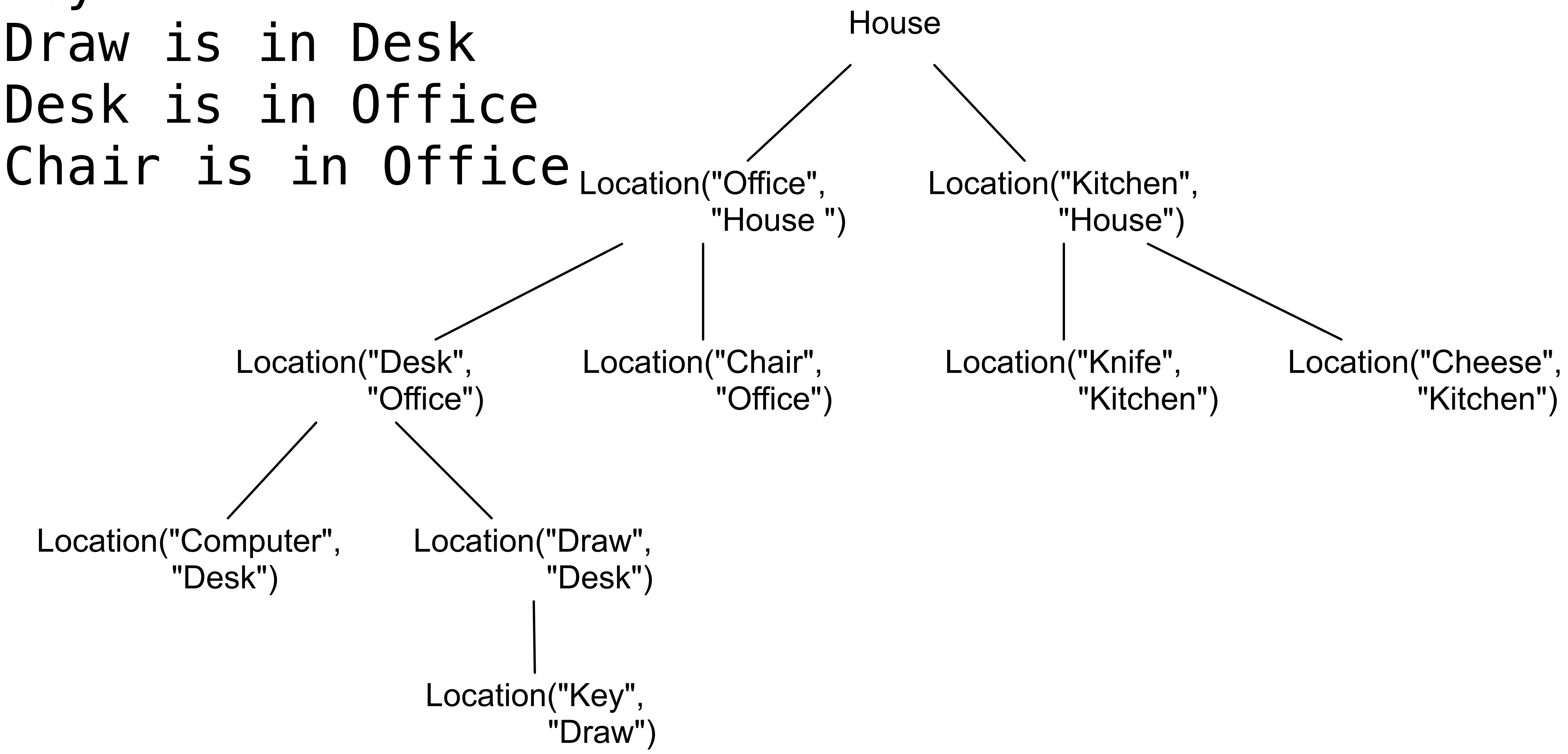

House
Location("Office", "House ")
Location("Kitchen", "House")
Location("Desk", "Office")
Location("Chair", "Office")
Location("Knife", "Kitchen")
Location("Cheese", "Kitchen")
Location("Computer", "Desk")
Location("Draw", "Desk")
Location("Key", "Draw")

Backward Chaining

```
rule "go5"
when
    String( this == "go5" )
    isContainedIn(thing, location; )
then
    System.out.println( "thing " + thing + " is in " + location );
end

ksession.insert( "go5" );
ksession.fireAllRules();
---
go5
thing Knife is in House
thing Cheese is in House
thing Key is in House
thing Computer is in House
thing Draw is in House
thing Desk is in House
thing Chair is in House
thing Key is in Office
thing Computer is in Office
thing Draw is in Office
thing Key is in Desk
thing Office is in House
thing Computer is in Desk
thing Knife is in Kitchen
thing Cheese is in Kitchen
thing Kitchen is in House
thing Key is in Draw
thing Draw is in Desk
thing Desk is in Office
thing Chair is in Office
```

Out Var (unbound)

Out Var (unbound)



House

Location("Office", "House ")

Location("Kitchen", "House")

Location("Desk", "Office")

Location("Chair", "Office")

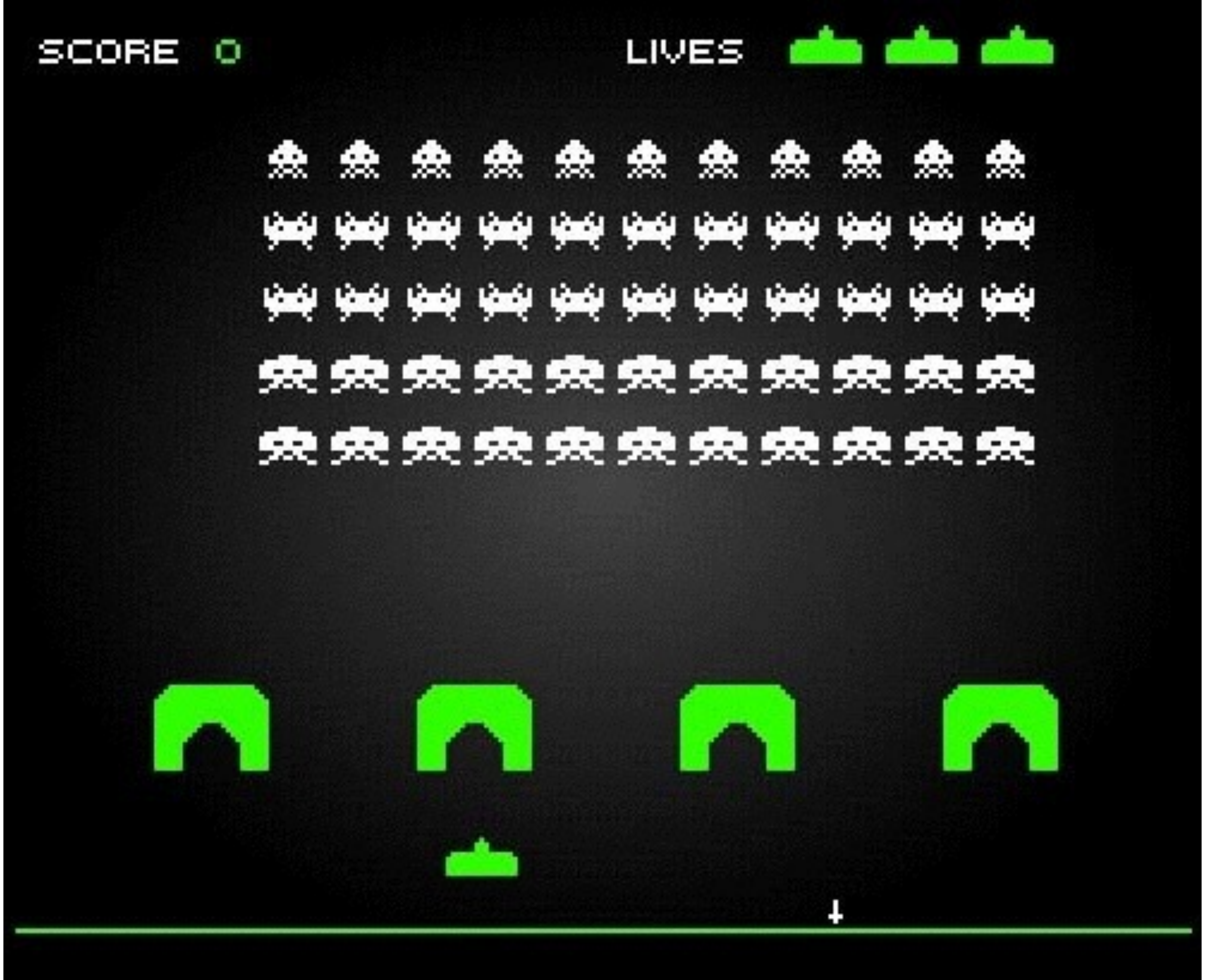
Location("Knife", "Kitchen")

Location("Cheese", "Kitchen")

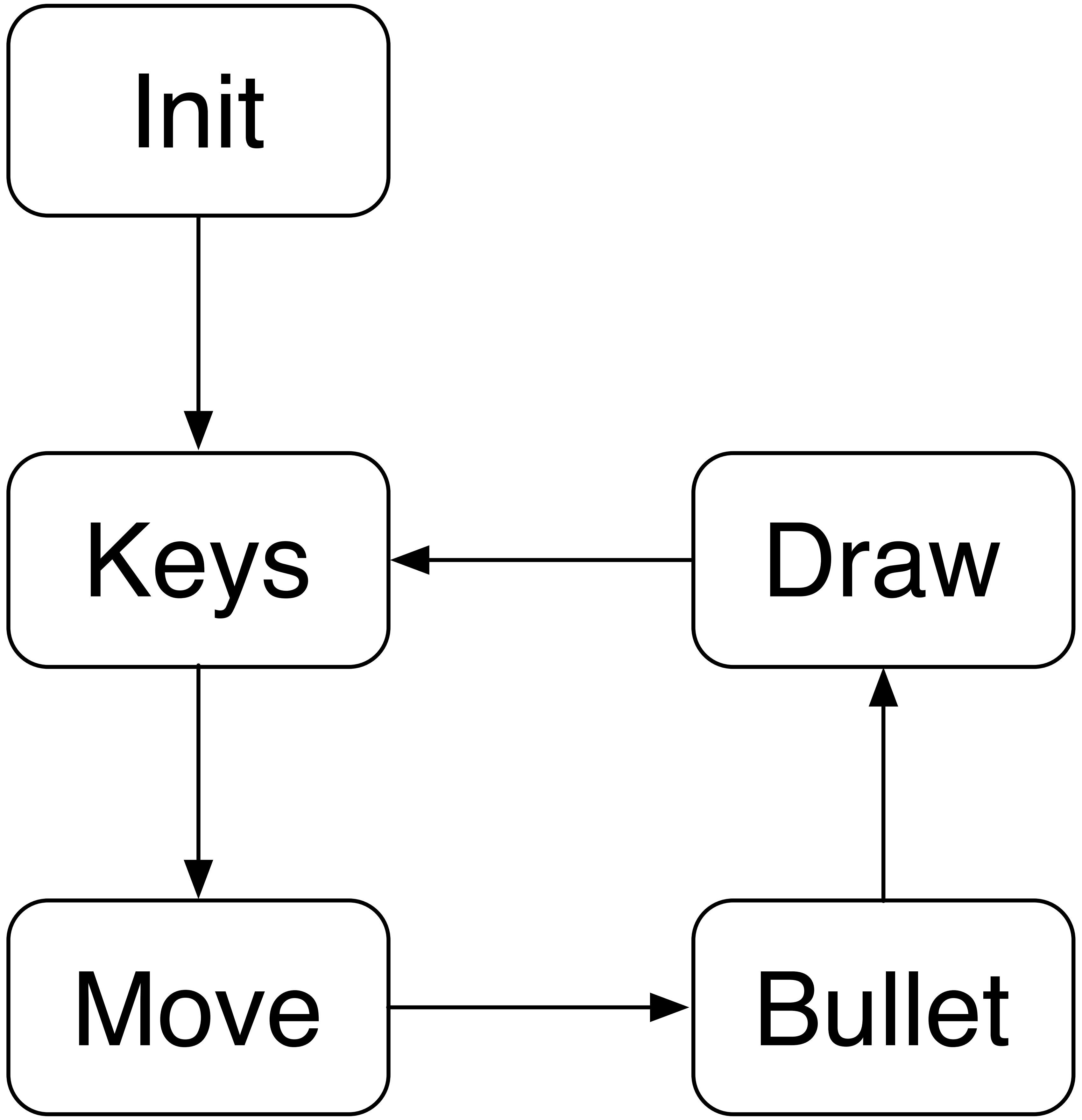
Location("Computer", "Desk")

Location("Draw", "Desk")

Location("Key", "Draw")


SCORE 0
LIVES

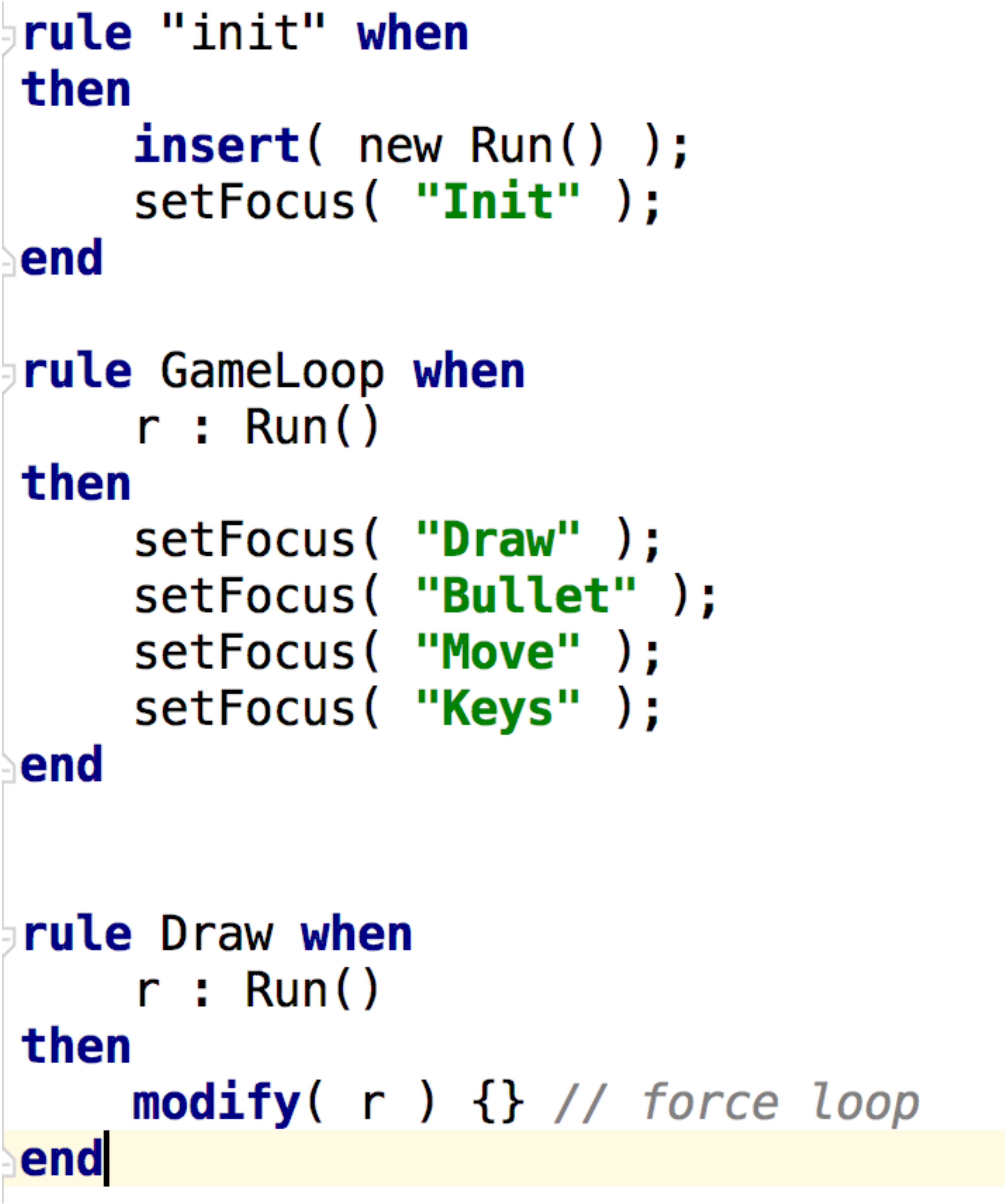
Invaders



Invaders

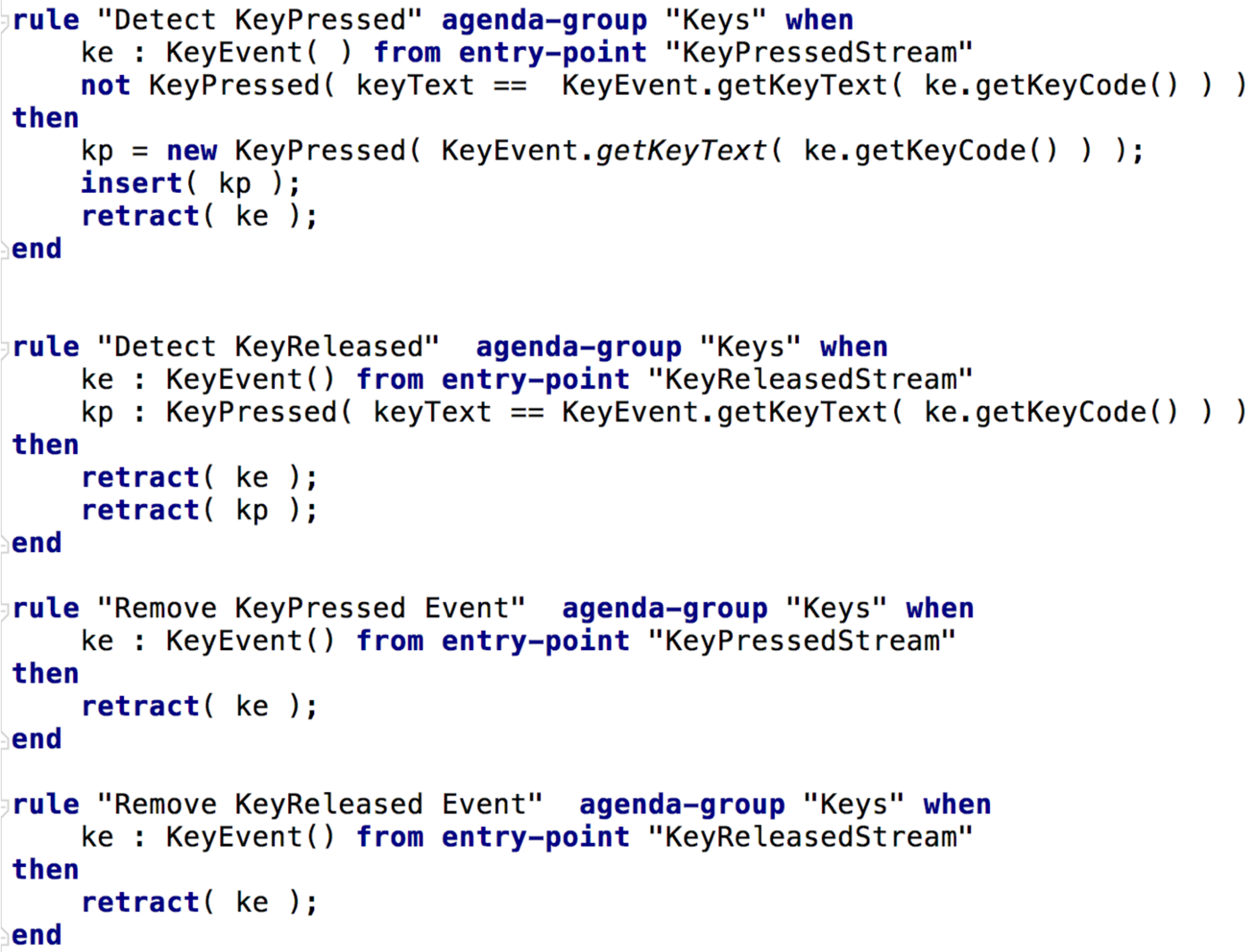
```
rule "init" when
then
    insert( new Run() );
    setFocus( "Init" );
end

rule GameLoop when
    r : Run()
then
    setFocus( "Draw" );
    setFocus( "Bullet" );
    setFocus( "Move" );
    setFocus( "Keys" );
end


rule Draw when
    r : Run()
then
    modify( r ) {} // force loop
end
```

Invaders



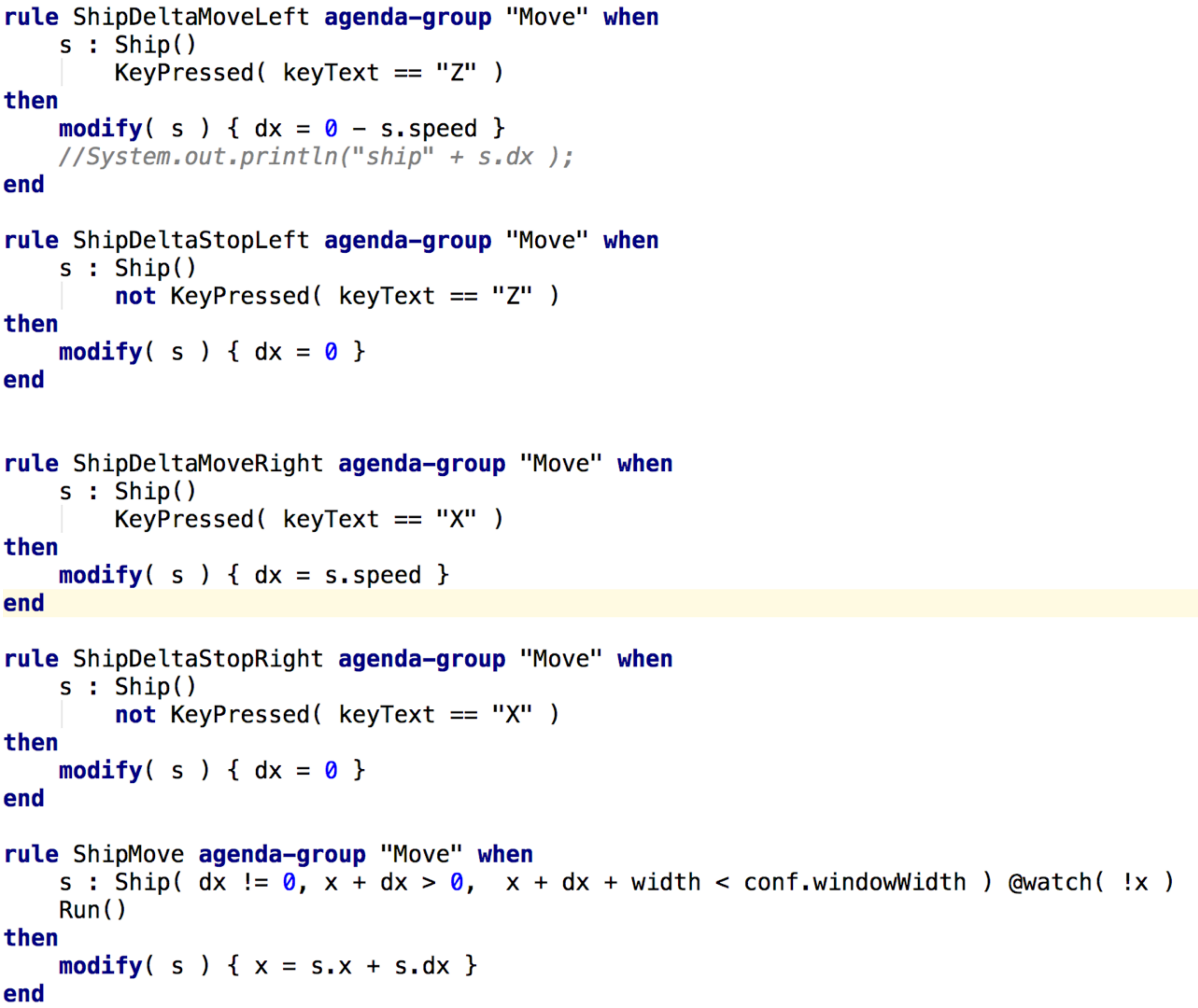
```
rule "init" when
then
    insert( new Run() );
    setFocus( "Init" );
end

rule GameLoop when
    r : Run()
then
    setFocus( "Draw" );
    setFocus( "Bullet" );
    setFocus( "Move" );
    setFocus( "Keys" );
end


rule Draw when
    r : Run()
then
    modify( r ) {} // force loop
end
```

Invaders



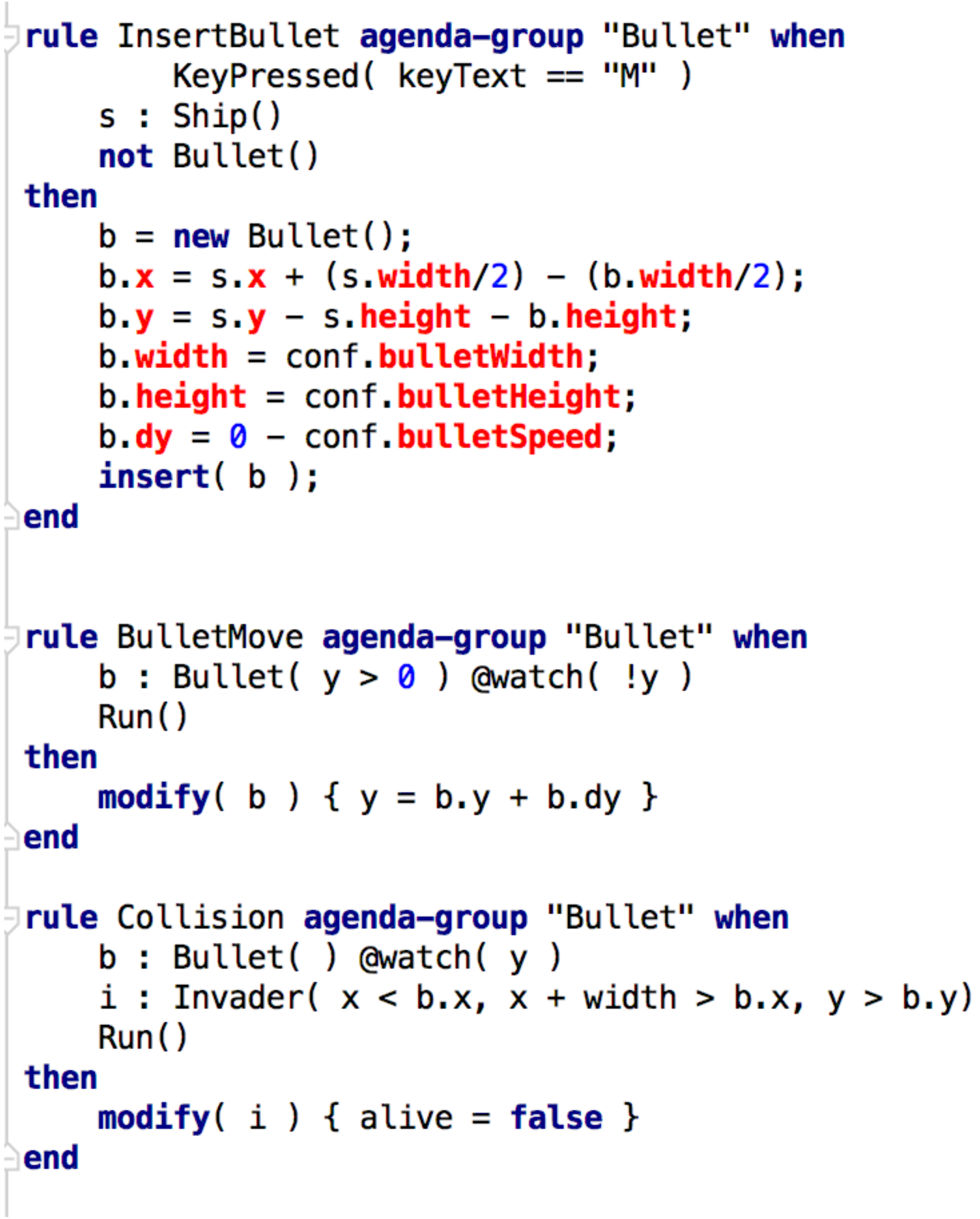
```
rule "Detect KeyPressed" agenda-group "Keys" when
    ke : KeyEvent( ) from entry-point "KeyPressedStream"
    not KeyPressed( keyText ==  KeyEvent.getKeyText( ke.getKeyCode() ) )
then
    kp = new KeyPressed( KeyEvent.getKeyText( ke.getKeyCode() ) );
    insert( kp );
    retract( ke );
end


rule "Detect KeyReleased"  agenda-group "Keys" when
    ke : KeyEvent() from entry-point "KeyReleasedStream"
    kp : KeyPressed( keyText == KeyEvent.getKeyText( ke.getKeyCode() ) )
then
    retract( ke );
    retract( kp );
end

rule "Remove KeyPressed Event"  agenda-group "Keys" when
    ke : KeyEvent() from entry-point "KeyPressedStream"
then
    retract( ke );
end

rule "Remove KeyReleased Event"  agenda-group "Keys" when
    ke : KeyEvent() from entry-point "KeyReleasedStream"
then
    retract( ke );
end
```

Invaders



```
rule ShipDeltaMoveLeft agenda-group "Move" when
    s : Ship()
        KeyPressed( keyText == "Z" )
then
    modify( s ) { dx = 0 - s.speed }
    //System.out.println("ship" + s.dx );
end

rule ShipDeltaStopLeft agenda-group "Move" when
    s : Ship()
        not KeyPressed( keyText == "Z" )
then
    modify( s ) { dx = 0 }
end


rule ShipDeltaMoveRight agenda-group "Move" when
    s : Ship()
        KeyPressed( keyText == "X" )
then
    modify( s ) { dx = s.speed }
end

rule ShipDeltaStopRight agenda-group "Move" when
    s : Ship()
        not KeyPressed( keyText == "X" )
then
    modify( s ) { dx = 0 }
end

rule ShipMove agenda-group "Move" when
    s : Ship( dx != 0, x + dx > 0,  x + dx + width < conf.windowWidth ) @watch( !x )
    Run()
then
    modify( s ) { x = s.x + s.dx }
end
```

Invaders



```
rule InsertBullet agenda-group "Bullet" when
        KeyPressed( keyText == "M" )
    s : Ship()
    not Bullet()
then
    b = new Bullet();
    b.x = s.x + (s.width/2) - (b.width/2);
    b.y = s.y - s.height - b.height;
    b.width = conf.bulletWidth;
    b.height = conf.bulletHeight;
    b.dy = 0 - conf.bulletSpeed;
    insert( b );
end


rule BulletMove agenda-group "Bullet" when
    b : Bullet( y > 0 ) @watch( !y )
    Run()
then
    modify( b ) { y = b.y + b.dy }
end

rule Collision agenda-group "Bullet" when
    b : Bullet( ) @watch( y )
    i : Invader( x < b.x, x + width > b.x, y > b.y)
    Run()
then
    modify( i ) { alive = false }
end
```

Invaders

```
rule ClearCanvas agenda-group "Draw"  salience 100 when
    Run()
then
   g = ui.getGraphics();
   g.setColor( Color.BLACK ); // background
   g.fillRect(0,0, conf.getWindowWidth(), conf.getWindowHeight() );
end

rule DrawShip agenda-group "Draw" when
  s : Ship()
      Run()
then
   g = ui.getGraphics();
   g.setColor( Color.BLACK ); // background
   g.fillRect( s.x - s.dx, s.y, s.width, s.height ); // restore the previous blackground
   g.drawImage( ImageIO.read( GameUI.class.getResource( "invaders/ship.gif" ) ), s.x, s.y, s.width, s.height, ui.getCanvas() );
end

rule DrawLiveInvader agenda-group "Draw" when
  i : Invader( alive == true)
      Run()
then
   g = ui.getGraphics();
   g.setColor( Color.BLACK ); // background
   g.drawImage( ImageIO.read( GameUI.class.getResource( "invaders/invader1.gif" ) ), i.x, i.y, i.width, i.height, ui.getCanvas() );
end
```