The Irish Software Engineering Research Center

Operational Semantics for KnowLang ASK and TELL Operators

Technical Report: LERO-TR2012.xx

Emil Vassev, University of Limerick, Ireland

Ver. 1.00

August 29th, 2012

1. Introduction

A system with KnowLang knowledge representation talks to the KnowLang Reasoner via a predefined set of TELL and ASK operators. TELL operators feed the KR context with important information driven by errors, executed actions, new sensory data, etc., thus helping the KnowLang Reasoner update the KR with recent changes in both the system and execution environment. The system uses ASK operators to receive recommended behavior where knowledge is used against the perception of the world to generate appropriate actions in compliance to some goals and beliefs. In addition ASK operators may provide the system with awareness-based conclusions about the current state of the system or the environment.

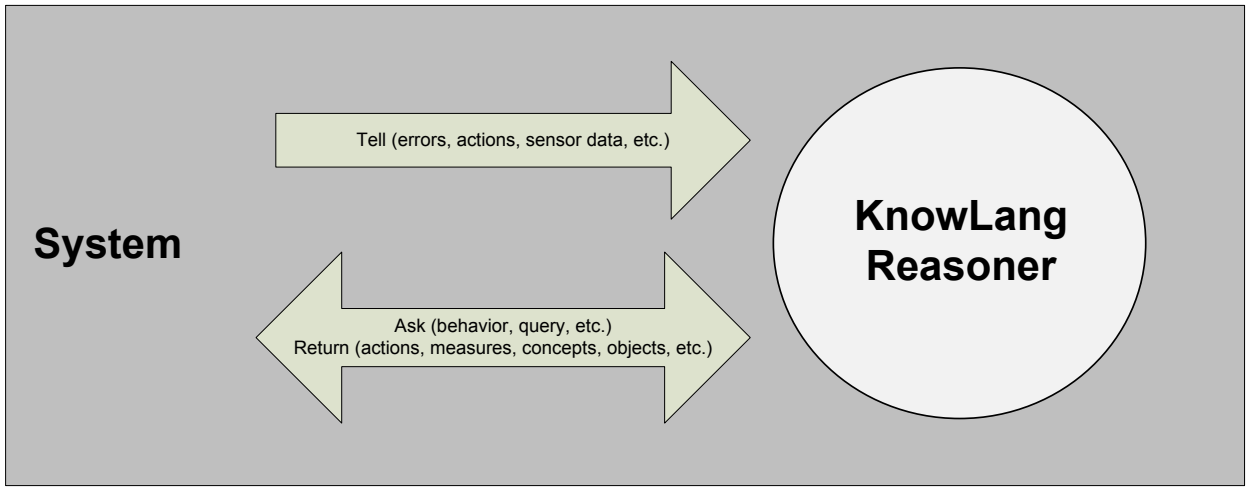
2. KnowLang ASK and TELL Operators

TELL

1. System TELL: error (error code, source), sensor data (source + data), action (class + method), update (class + properties or variable)
2. Reasoner: map input to KR Symbols
3. Reasoner: update KB (add new objects and concepts and change states)

ASK

1. System ASK: behavior, rule-based behavior, current state and current situation
2. Reasoner: map input to KR Symbols
3. Reasoner: get behavior actions, retrieve information, update KB
4. Reasoner: return result to system



3. Operational Semantics for TELL Operators

$\sigma - system\ operational\ (SO)\ context;\ \sigma' - system\ KB\ context$

Tell_Err – Tell the Reasoner about a Raised Error

1) $$\frac{\sigma \xrightarrow{tell\_err(er,s)} \sigma'}{\langle TELL\_ERR\ (er,s), \sigma' \rangle \rightarrow \langle findErrorConcept(er,\ s), \sigma' \rangle}$$ *er-error in SO context; s-error source string*

2) $$\frac{\sigma \xrightarrow{tell\_err(er,s)} \sigma'\ \langle findErrorConcept(er,\ s), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}{\langle\ createErrObj(c,er,s), \sigma' \rangle \rightarrow \langle o_{err}, \sigma' \rangle}$$

3) $$\frac{\sigma \xrightarrow{tell\_err(er,s)} \sigma'\ \langle findErrorConcept(er,\ s), \sigma' \rangle \rightarrow \langle NIL, \sigma' \rangle}{\langle\ createErrConcept(er,s), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}$$

4) $$\frac{\sigma \xrightarrow{tell\_err(er,s)} \sigma'\ \langle\ createErrConcept(er,s), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}{\langle\ createErrObj(c,er,s), \sigma' \rangle \rightarrow \langle o_{err}, \sigma' \rangle}$$ *c-error concept in KB; o_{err}-error object in KB*

5) $$\frac{\langle createErrObj(c,er,s), \sigma' \rangle \rightarrow \langle o_{err}, \sigma' \rangle\ \ \langle findSourceObject(s), \sigma' \rangle \rightarrow \langle o_s, \sigma' \rangle}{\langle setCurrentState(o_s,o_{err}), \sigma' \rangle \rightarrow \langle o_s.STATE, \sigma' \rangle}$$ *o_s -source object in KB*

***KnowLang Grammar**:*
State-Body | Bln-Expr
Bln-Reln | RAISED (Error-Name)

6) $$\frac{\langle setCurrentState(o_1,o_2), \sigma' \rangle \rightarrow \langle o_1.STATE, \sigma' \rangle}{\langle findPropertyHosts(o_1), \sigma' \rangle \rightarrow \langle O_h, \sigma' \rangle}$$

7) $$\frac{\langle setCurrentState(o_1,o_2), \sigma' \rangle \rightarrow \langle o_1.STATE, \sigma' \rangle\ \langle findPropertyHosts(o_1), \sigma' \rangle \rightarrow \langle O_h, \sigma' \rangle}{\forall\ o_h \in O_h \bullet \langle setCurrentState(o_h,o_1), \sigma' \rangle \rightarrow \langle o_h.STATE, \sigma' \rangle}$$

8) $$\frac{\langle createErrObj(c,er,s), \sigma' \rangle \rightarrow \langle o, \sigma' \rangle\ \langle findErrEvents(c), \sigma' \rangle \rightarrow \langle E_{err}, \sigma' \rangle}{\forall\ e_{err} \in E_{err} \bullet \langle fireEvent(\ e_{err}), \sigma' \rangle \rightarrow \langle o_e, \sigma' \rangle}$$

***KnowLang Grammar**:*
o_e-event instance (fired event)
Error-Activation | RAISED { Error-Name }

9) $$\frac{\langle fireEvent(e), \sigma' \rangle \rightarrow \langle o_e, \sigma' \rangle\ \langle findDependedObjects(e), \sigma' \rangle \rightarrow \langle O_d, \sigma' \rangle}{\forall\ o_d \in O_d \bullet \langle setCurrentState(o_d,e), \sigma' \rangle \rightarrow \langle o_d.STATE, \sigma' \rangle}$$

***KnowLang Grammar**:*
State-Body | Bln-Expr
Bln-Reln | OCCURRED (Event-Name)

10) $$\frac{\langle fireEvent(e), \sigma' \rangle \rightarrow \langle o_e, \sigma' \rangle\ \forall\ o_d \in O_d \bullet \langle setCurrentState(o_d,e), \sigma' \rangle \rightarrow \langle o_d.STATE, \sigma' \rangle}{\langle findCurrentSituation(), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}$$

11) $$\frac{\langle fireEvent(e), \sigma' \rangle \rightarrow \langle o_e, \sigma' \rangle\ \langle findCurrentSituation() \rightarrow \langle si, \sigma' \rangle}{\langle recordEventHistory(o_e,si), \sigma' \rangle \rightarrow \langle E_{si}^{\leftarrow}, \sigma' \rangle}$$

Tell_Sensor – Tell the Reasoner about new Data Detected by a Sensor

12) $\frac{\sigma \xrightarrow{tell\_sensor(d,s)} \sigma'}{\langle TELL\_SENSOR(d,s), \sigma' \rangle \rightarrow \langle findMetricConcept(s), \sigma' \rangle}$ *d-data, s-source(driver class, instance, method, parameters)*

13) $\frac{\sigma \xrightarrow{tell\_sensor\ (d,s)} \sigma' \ \langle findMetricConcept(s), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}{\langle\ findMetricObject(c,s), \sigma' \rangle \rightarrow \langle o_m, \sigma' \rangle}$ *c-metric concept, o_m-metric object*

14) $\frac{\sigma \xrightarrow{tell\_sensor\ (d,s)} \sigma' \ \langle findMetricObject(c,s), \sigma' \rangle \rightarrow \langle o_m, \sigma' \rangle}{\langle update(o_m, d), \sigma' \rangle \rightarrow \langle o_m\prime, \sigma' \rangle}$ *o_m'–updated metric object*

15) $\frac{\langle update(o_m, d), \sigma' \rangle \rightarrow \langle o_m\prime, \sigma' \rangle \ \langle findMetricEvents(o_m\prime), \sigma' \rangle \rightarrow \langle E_m, \sigma' \rangle}{\forall\ e_m \in E_m \bullet \langle fireEvent(\ e_m), \sigma' \rangle \rightarrow \langle o_e, \sigma' \rangle}$ *o_e-event instance (fired event)*

KnowLang Grammar:
Metric-Activation | CHANGED { Metric-Name }

Tell_Action – Tell the Reasoner about Action execution

16) $$\frac{\sigma \xrightarrow{tell\_action(P,s)} \sigma'}{\langle TELL\_ACTION\ (P,s), \sigma' \rangle \rightarrow \langle findActionConcept(P,s), \sigma' \rangle}$$ *P-parameters, s-source(class, instance, method)*

17) $$\frac{\sigma \xrightarrow{tell\_action(P,s)} \sigma' \ \langle findActionConcept(P,s), \sigma' \rangle \rightarrow \langle a, \sigma' \rangle}{\langle executeAction(a), \sigma' \rangle \rightarrow \langle o_a, \sigma' \rangle}$$

18) $$\frac{\langle executeAction(a), \sigma' \rangle \rightarrow \langle o_a, \sigma' \rangle}{\langle findDependedObjects(a), \sigma' \rangle \rightarrow \langle O_d, \sigma' \rangle}$$

19) $$\frac{\langle executeAction(a), \sigma' \rangle \rightarrow \langle o_a, \sigma' \rangle \ \langle findDependedObjects(a), \sigma' \rangle \rightarrow \langle O_d, \sigma' \rangle}{\forall\ o_d \in O_d \bullet \langle setCurrentState(o_d, e), \sigma' \rangle \rightarrow \langle o_d.STATE, \sigma' \rangle}$$

KnowLang Grammar:
State-Body | Bln-Expr
Bln-Reln | EXECUTED (Action-Name)

20) $$\frac{\langle executeAction(a), \sigma' \rangle \rightarrow \langle o_a, \sigma' \rangle \ \ \forall\ o_d \in O_d \bullet \langle setCurrentState(o_d, e), \sigma' \rangle \rightarrow \langle o_d.STATE, \sigma' \rangle}{\langle findCurrentSituation(), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}$$

21) $$\frac{\langle executeAction(a), \sigma' \rangle \rightarrow \langle o_a, \sigma' \rangle \ \langle findCurrentSituation(), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}{\langle recordActionHistory(o_a, si), \sigma' \rangle \rightarrow \langle a_{si}^{\leftarrow}, \sigma' \rangle}$$

Tell_Action (behavior) – Tell about Action Execution as part of behavior performance

22) $$\frac{\sigma \xrightarrow{tell\_action(b_{si}^{\pi},P,s)} \sigma'}{\langle TELL\_ACTION\ (P,s), \sigma' \rangle \rightarrow \langle findActionConcept(P,s), \sigma' \rangle}$$ *P-parameters, s-source(class, instance, method)*

23) $$\frac{\sigma \xrightarrow{tell\_action(b_{si}^{\pi},P,s)} \sigma' \ \langle findActionConcept(P,s), \sigma' \rangle \rightarrow \langle a, \sigma' \rangle}{\langle executeAction(a), \sigma' \rangle \rightarrow \langle o_a, \sigma' \rangle}$$

24) $$\frac{\sigma \xrightarrow{tell\_action(b_{si}^{\pi},P,s)} \sigma'}{\langle RETRIEVE\_BEHAVIOR\ (b_{si}^{\pi}), \sigma' \rangle \rightarrow \langle b_{si}^{\pi}, \sigma' \rangle}$$

25) $$\frac{\langle executeAction(a), \sigma' \rangle \rightarrow \langle o_a, \sigma' \rangle \ \ \langle RETRIEVE\_BEHAVIOR\ (b_{si}^{\pi}), \sigma' \rangle \rightarrow \langle b_{si}^{\pi}, \sigma' \rangle}{\langle recordBehaviorHistory\ (b_{si}^{\pi}, o_a), \sigma' \rangle \rightarrow \langle b_{si}^{\pi}, \sigma' \rangle}$$

26) $$\frac{\langle recordBehaviorHistory\ (b_{si}^{\pi}, o_a), \sigma' \rangle \rightarrow \langle b_{si}^{\pi}, \sigma' \rangle \ \langle isLastAction(b_{si}^{\pi}, o_a), \sigma' \rangle \rightarrow \langle TRUE, \sigma' \rangle}{\langle getGoalState\ (b_{si}^{\pi}), \sigma' \rangle \rightarrow \langle st, \sigma' \rangle}$$

27) $$\frac{\langle isLastAction(b_{si}^{\pi}, o_a), \sigma' \rangle \rightarrow \langle TRUE, \sigma' \rangle \ \langle getGoalState\ (b_{si}^{\pi}), \sigma' \rangle \rightarrow \langle st, \sigma' \rangle \ \langle getCurrState\ (), \sigma' \rangle \rightarrow \langle st_s, \sigma' \rangle \langle findSitnPolcyRltns(b_{si}^{\pi}), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle}{\langle recomputeProbabilityDistr\ (Rl_{si}, st, st_s), \sigma' \rangle \rightarrow \langle Rl'_{si}, \sigma' \rangle}$$

28) $$\frac{\langle isLastAction(b_{si}^{\pi}, o_a), \sigma' \rangle \rightarrow \langle TRUE, \sigma' \rangle \ \langle getGoalState\ (b_{si}^{\pi}), \sigma' \rangle \rightarrow \langle st, \sigma' \rangle \ \langle getCurrState\ (), \sigma' \rangle \rightarrow \langle st_s, \sigma' \rangle \langle findPolcyMappings(b_{si}^{\pi}), \sigma' \rangle \rightarrow \langle M_{\pi}, \sigma' \rangle}{\langle recomputeProbabilityDistr\ (M_{\pi}, st, st_s), \sigma' \rangle \rightarrow \langle M'_{\pi}, \sigma' \rangle}$$

Tell_Obj_Update – Tell the Reasoner to update an object

29) $$\frac{\sigma \xrightarrow{tell\_obj\_update(<Pr,D>,s)} \sigma'}{\langle TELL\_OBJ\_UPDATE\ (<Pr,D>,s), \sigma' \rangle \rightarrow \langle findKRObject(s), \sigma' \rangle}$$

Pr *-properties to be updated;* ***D*** *-new data;* ***s*** *-implementation string*

30) $$\frac{\sigma \xrightarrow{tell\_obj\_update(<Pr,D>,s)} \sigma' \ \langle findKRObject(s), \sigma' \rangle \rightarrow \langle o, \sigma' \rangle}{\langle update(o,<Pr,D>), \sigma' \rangle \rightarrow \langle o, \sigma' \rangle}$$

31) $$\frac{\sigma \xrightarrow{tell\_obj\_update(<Pr,D>,s)} \sigma' \ \langle update(o,<Pr,D>), \sigma' \rangle \rightarrow \langle o, \sigma' \rangle}{\langle setCurrentState(o), \sigma' \rangle \rightarrow \langle o.STATE, \sigma' \rangle}$$

32) $$\frac{\langle setCurrentState(o), \sigma' \rangle \rightarrow \langle o.STATE, \sigma' \rangle}{\langle findPropertyHosts(o), \sigma' \rangle \rightarrow \langle O_h, \sigma' \rangle}$$

33) $$\frac{\langle setCurrentState(o), \sigma' \rangle \rightarrow \langle o.STATE, \sigma' \rangle \ \langle findPropertyHosts(o), \sigma' \rangle \rightarrow \langle O_h, \sigma' \rangle}{\forall\, o_h \in O_h \bullet \langle setCurrentState(o_h,o), \sigma' \rangle \rightarrow \langle o_h.STATE, \sigma' \rangle}$$

Tell_Cncpt_Update – Tell the Reasoner to update a concept

34) $$\frac{\sigma \xrightarrow{tell\_cncpt\_update(<Pr,D>,s)} \sigma'}{\langle TELL\_CNCPT\_UPDATE\ (<Pr,D>,s), \sigma' \rangle \rightarrow \langle findKRConcept(s), \sigma' \rangle}$$

Pr *-properties to be updated;* ***D*** *-new data;* ***s*** *-implementation string*

35) $$\frac{\sigma \xrightarrow{tell\_cncpt\_update(<Pr,D>,s)} \sigma' \ \langle findKRConcept(s), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}{\langle update(c,<Pr,D>), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}$$

36) $$\frac{\sigma \xrightarrow{tell\_cncpt\_update(<Pr,D>,s)} \sigma' \ \langle update(c,<Pr,D>), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}{\langle findConceptObjects(c), \sigma' \rangle \rightarrow \langle O_c, \sigma' \rangle}$$

37) $$\frac{\sigma \xrightarrow{tell\_cncpt\_update(<Pr,D>,s)} \sigma' \ \langle update(c,<Pr,D>), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle \ \langle findConceptObjects(c), \sigma' \rangle \rightarrow \langle O_c, \sigma' \rangle}{\forall\, o_c \in O_c \bullet \langle update(o_c,<Pr,D>), \sigma' \rangle \rightarrow \langle o_c, \sigma' \rangle}$$

38) $$\frac{\langle update(o,<Pr,D>), \sigma' \rangle \rightarrow \langle o, \sigma' \rangle}{\langle setCurrentState(o), \sigma' \rangle \rightarrow \langle o.STATE, \sigma' \rangle}$$

39) $$\frac{\langle setCurrentState(o), \sigma' \rangle \rightarrow \langle o.STATE, \sigma' \rangle}{\langle findPropertyHosts(o), \sigma' \rangle \rightarrow \langle O_h, \sigma' \rangle}$$

40) $$\frac{\langle setCurrentState(o), \sigma' \rangle \rightarrow \langle o.STATE, \sigma' \rangle \ \langle findPropertyHosts(o), \sigma' \rangle \rightarrow \langle O_h, \sigma' \rangle}{\forall\, o_h \in O_h \bullet \langle setCurrentState(o_h,o), \sigma' \rangle \rightarrow \langle o_h.STATE, \sigma' \rangle}$$

4. Operational Semantics for ASK Operators

$\sigma - system\ operational\ (SO)\ context;\ \sigma' - system\ KB\ context$

Ask_Behavior – Ask for self-adaptive behavior

41) $$\frac{\sigma \xrightarrow{ask\_behavior()} \sigma'}{\langle ASK\_BEHAVIOR, \sigma' \rangle \rightarrow \langle findCurrentSituation(), \sigma' \rangle}$$

42) $$\frac{\sigma \xrightarrow{ask\_behavior()} \sigma' \ \langle findCurrentSituation(), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}{\langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle}$$

43) $$\frac{\sigma \xrightarrow{ask\_behavior()} \sigma' \ \langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle}{\langle \max(Rl_{si}), \sigma' \rangle \rightarrow \langle \pi_{si}, \sigma' \rangle}$$ *policy with the highest probability is considered*

44) $\langle \pi, \sigma' \rangle \rightarrow \langle applyPolicy(\pi), \sigma' \rangle$ *evaluate policy* $\boldsymbol{\pi}$ *in context* $\boldsymbol{\sigma'}$

45) $$\frac{\langle \pi_{si}, \sigma' \rangle \rightarrow \langle applyPolicy(\pi_{si}), \sigma' \rangle \ \forall n_\pi \in N_\pi \bullet \langle n_\pi, \sigma' \rangle \rightarrow \langle TRUE, \sigma' \rangle}{\langle map(\pi_{si}, N_\pi, A_\pi, Pr), \sigma' \rangle \rightarrow \langle <A'_\pi, Pr'>, \sigma' \rangle} \ A'_\pi \subseteq A_\pi$$

Condition-Action-Mapping | MAPPING { CONDITIONS { Bln-Expr } DO_ACTIONS { Action-Names } PROBABILITY { Probability-Number }}

46) $$\frac{\langle \pi_{si}, \sigma' \rangle \rightarrow \langle applyPolicy(\pi_{si}), \sigma' \rangle \ \langle map(\pi_{si}, N_\pi, A_\pi, Pr), \sigma' \rangle \rightarrow \langle <A'_\pi, Pr'>, \sigma' \rangle \ \langle \max(Pr'), \sigma' \rangle \rightarrow \langle pr, \sigma' \rangle}{\langle getProbableActions(<A'_\pi, Pr'>, pr), \sigma' \rangle \rightarrow \langle <A''_\pi, pr>, \sigma' \rangle} \ A''_\pi \subseteq A'_\pi$$

47) $$\frac{\langle \pi_{si}, \sigma' \rangle \rightarrow \langle applyPolicy(\pi_{si}), \sigma' \rangle \ \langle map(\pi_{si}, N_\pi, A_\pi, Pr), \sigma' \rangle \rightarrow \langle <A'_\pi, Pr'>, \sigma' \rangle \ \langle getProbableActions(<A'_\pi, Pr'>, pr), \sigma' \rangle \rightarrow \langle <A''_\pi, pr>, \sigma' \rangle}{\langle recordBehavior(\pi_{si}, A''_\pi), \sigma' \rangle \rightarrow \langle b^\pi_{si}, \sigma' \rangle} \ A''_\pi \subseteq A_{si}$$

48) $$\frac{\sigma \xrightarrow{ask\_behavior()} \sigma' \ \langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^\pi_{si}, \sigma' \rangle}{\sigma' \xrightarrow{return(b^\pi_{si})} \sigma} \ A_\pi \subseteq A_{si}$$

Ask_Behavior (goal) – Ask for self-adaptive behavior to achieve certain goal

Note: The goal is pursued from the current situation, i.e., the situation must be related to a policy (one or more) having this goal in order to generate a behavior.

49) $$\frac{\sigma \xrightarrow{ask\_behavior(g)} \sigma'}{\langle ASK\_BEHAVIOR(g), \sigma' \rangle \rightarrow \langle findCurrentSituation(), \sigma' \rangle}$$

50) $$\frac{\sigma \xrightarrow{ask\_behavior(g)} \sigma' \ \langle findCurrentSituation(), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}{\langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle}$$

51) $$\frac{\sigma \xrightarrow{ask\_behavior(g)} \sigma' \ \langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findGoalPolicy(Rl_{si}, g), \sigma' \rangle \rightarrow \langle \pi_{si}, \sigma' \rangle}{\langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}$$

$findGoalPolicy(Rl_{si}, g)$ *finds a policy with: 1)desired goal; and 2)highest probability; among the policies related to situation* ***si***

52) $$\frac{\sigma \xrightarrow{ask\_behavior(g)} \sigma' \ \langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}{\sigma' \xrightarrow{return\left(b^{\pi}_{si}\right)} \sigma} \ A_\pi \subseteq A_{si}$$

53) $$\frac{\sigma \xrightarrow{ask\_behavior(g)} \sigma' \ \langle findSitnPolcyRltns(si_c), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findGoalPolicy(Rl_{si}, g), \sigma' \rangle \rightarrow \langle NIL, \sigma' \rangle}{\langle findClosestStateSituation(si_c, g), \sigma' \rangle \rightarrow \langle \langle \overrightarrow{Si_{si}}, G_\pi \rangle, si' \rangle, \sigma' \rangle}$$ *no policy was found that can pursue the desired goal* ***g*** *from situation* $\boldsymbol{si_c}$ *(current situation)*

$findClosestStateSituation(si, g)$ *finds a situation: 1) from where we can pursue the* ***g*** *goal; and 2) is the "closest" to situation* ***si;*** *the search is to be done in the situations concept map where situations are nodes and the edges are policy goals relating those situations;*

54) $$\frac{\sigma \xrightarrow{ask\_behavior(g_i)} \sigma' \ \langle findClosestStateSituation(si_c, g_i), \sigma' \rangle \rightarrow \langle \langle \overrightarrow{Si_{si}}, G_\pi \rangle, si' \rangle, \sigma' \rangle}{\forall \langle si, g \rangle \in \langle \overrightarrow{Si_{si}}, G_\pi \rangle \bullet \langle ASK\_BEHAVIOR(si, g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle}$$

55) $$\frac{\langle ASK\_BEHAVIOR(si, g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findGoalPolicy(Rl_{si}, g), \sigma' \rangle \rightarrow \langle \pi_{si}, \sigma' \rangle}{\langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}$$

56) $$\frac{\sigma \xrightarrow{ask\_behavior(g_i)} \sigma' \ \langle ASK\_BEHAVIOR(si, g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}{\langle addBehavior\left(b^{\pi}_{si}\right), \sigma' \rangle \rightarrow \langle \overrightarrow{B^{\pi}_{si}}, \sigma' \rangle}$$

$$\sigma \xrightarrow{ask\_behavior(g_i)} \sigma' \ \langle findClosestStateSituation(si_c, g_i), \sigma' \rangle \rightarrow \langle \ll \overrightarrow{Si_{si}}, G_\pi \rangle, si' \rangle, \sigma' \rangle$$

57) $$\frac{\langle ASK\_BEHAVIOR(si, g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle addBehavior\left(b^{\pi}_{si}\right), \sigma' \rangle \rightarrow \langle \overrightarrow{B^{\pi}_{si}}, \sigma' \rangle}{\sigma' \xrightarrow{return\left(\overrightarrow{B^{\pi}_{si}}\right)} \sigma} \ si' = si \wedge g = g_i$$ *return summarized behavior if the last processed situation is the* ***closest*** *to the initial* si_c

Ask_Behavior (situation, goal) – Ask for self-adaptive behavior to achieve certain goal when departing from a specific situation

58) $$\frac{\sigma \xrightarrow{ask\_behavior(si,g)} \sigma'}{\langle ASK\_BEHAVIOR(si,g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle}$$

59) $$\frac{\sigma \xrightarrow{ask\_behavior(si,g)} \sigma' \ \langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findGoalPolicy(Rl_{si}, g), \sigma' \rangle \rightarrow \langle \pi_{si}, \sigma' \rangle}{\langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}$$

60) $$\frac{\sigma \xrightarrow{ask\_behavior(si,g)} \sigma' \ \langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}{\sigma' \xrightarrow{return\left(b^{\pi}_{si}\right)} \sigma} \ A_\pi \subseteq A_{si}$$

61) $$\frac{\sigma \xrightarrow{ask\_behavior(si_i,g)} \sigma' \ \langle findSitnPolcyRltns(si_i), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findGoalPolicy(Rl_{si}, g), \sigma' \rangle \rightarrow \langle NIL, \sigma' \rangle}{\langle findClosestStateSituation(si_i, g), \sigma' \rangle \rightarrow \langle \langle \overrightarrow{Si_{si_i}}, G_\pi \rangle, si' \rangle, \sigma' \rangle}$$ *no policy was found that can pursue the desired goal* ***g*** *from situation* $\boldsymbol{si_i}$ *(initial situation)*

$findClosestStateSituation(si, g)$ *finds a situation: 1) from where we can pursue the* ***g*** *goal; and 2) is the "closest" to situation* ***si;*** *the search is to be done in the situations concept map where situations are nodes and the edges are policy goals relating those situations;*

62) $$\frac{\sigma \xrightarrow{ask\_behavior(si_i,g_i)} \sigma' \ \langle findClosestStateSituation(si_i, g_i), \sigma' \rangle \rightarrow \langle \langle \overrightarrow{Si_{si_i}}, G_\pi \rangle, si' \rangle, \sigma' \rangle}{\forall \langle si,g \rangle \in \langle \overrightarrow{Si_{si_i}}, G_\pi \rangle \bullet \langle ASK\_BEHAVIOR(si,g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle}$$

63) $$\frac{\langle ASK\_BEHAVIOR(si,g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findGoalPolicy(Rl_{si}, g), \sigma' \rangle \rightarrow \langle \pi_{si}, \sigma' \rangle}{\langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}$$

64) $$\frac{\sigma \xrightarrow{ask\_behavior(si_i,g_i)} \sigma' \ \langle ASK\_BEHAVIOR(si,g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}{\langle addBehavior\left(b^{\pi}_{si}\right), \sigma' \rangle \rightarrow \langle \overrightarrow{B^{\pi}_{si}}, \sigma' \rangle}$$

65) $$\frac{\begin{array}{c}\sigma \xrightarrow{ask\_behavior(si_i,g_i)} \sigma' \ \langle findClosestStateSituation(si_i, g_i), \sigma' \rangle \rightarrow \langle \langle\langle \overrightarrow{Si_{si_i}}, G_\pi \rangle, si' \rangle, \sigma' \rangle \\ \langle ASK\_BEHAVIOR(si,g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle addBehavior(b^{\pi}_{si}), \sigma' \rangle \rightarrow \langle \overrightarrow{B^{\pi}_{si}}, \sigma' \rangle\end{array}}{\sigma' \xrightarrow{return\left(\overrightarrow{B^{\pi}_{si}}\right)} \sigma}$$ $si' = si$ *return summarized behavior if the last processed situation is the* ***closest*** *to the initial* si_c

Ask_Behavior (state) – Ask for Self-adaptive Behavior to go to a certain state

66) $$\frac{\sigma \xrightarrow{ask\_behavior(s)} \sigma'}{\langle ASK\_BEHAVIOR(s), \sigma' \rangle \rightarrow \langle findCurrentSituation(), \sigma' \rangle}$$

67) $$\frac{\sigma \xrightarrow{ask\_behavior(s)} \sigma' \ \langle findCurrentSituation(), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}{\langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle}$$

68) $$\frac{\sigma \xrightarrow{ask\_behavior(s)} \sigma' \ \langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findStatePolicy(Rl_{si}, s), \sigma' \rangle \rightarrow \langle \pi_{si}, \sigma' \rangle}{\langle findStatePolicy(Rl_{si}, s), \sigma' \rangle \rightarrow \langle \pi_{si}, \sigma' \rangle}$$

$findStatePolicy(Rl_{si}, s)$ *finds policy with: 1)desired goal state; and 2)highest probability;*

69) $$\frac{\sigma \xrightarrow{ask\_behavior(s)} \sigma' \ \langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}{\sigma' \xrightarrow{return\left(b^{\pi}_{si}\right)} \sigma} \ A_\pi \subseteq A_{si}$$

70) $$\frac{\sigma \xrightarrow{ask\_behavior(s)} \sigma' \ \langle findSitnPolcyRltns(si_c), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findStatePolicy(Rl_{si}, s), \sigma' \rangle \rightarrow \langle NIL, \sigma' \rangle}{\langle findClosestStateSituation(si_c, s), \sigma' \rangle \rightarrow \langle <<Si^{\rightarrow}_{si,}, G_\pi>, si'>, \sigma' \rangle}$$ *no policy was found that can lead to the desired state* ***s*** *from situation* $\boldsymbol{si_c}$ *(current situation)*

$findClosestStateSituation(si, s)$ *finds a situation that has a state* ***s*** *and is the "closest" to situation* ***si;*** *the search is to be done in the situations concept map where situations are nodes and the edges are policy goals relating those situations;*

71) $$\frac{\sigma \xrightarrow{ask\_behavior(s)} \sigma' \ \langle findClosestStateSituation(si_c, s), \sigma' \rangle \rightarrow \langle <<Si^{\rightarrow}_{si,}, G_\pi>, si'>, \sigma' \rangle}{\forall <si, g> \in <Si^{\rightarrow}_{si,}, G_\pi> \bullet \langle ASK\_BEHAVIOR(si, g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle}$$

72) $$\frac{\langle ASK\_BEHAVIOR(si, g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle findSitnPolcyRltns(si), \sigma' \rangle \rightarrow \langle Rl_{si}, \sigma' \rangle \ \langle findGoalPolicy(Rl_{si}, g), \sigma' \rangle \rightarrow \langle \pi_{si}, \sigma' \rangle}{\langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}$$

73) $$\frac{\sigma \xrightarrow{ask\_behavior(s)} \sigma' \ \langle ASK\_BEHAVIOR(si, g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle recordBehavior(\pi_{si}, A_\pi), \sigma' \rangle \rightarrow \langle b^{\pi}_{si}, \sigma' \rangle}{\langle addBehavior\left(b^{\pi}_{si}\right), \sigma' \rangle \rightarrow \langle \overrightarrow{B^{\pi}_{si}}, \sigma' \rangle}$$

74) $$\frac{\sigma \xrightarrow{ask_{behavior(s)}} \sigma' \ \langle findClosestStateSituation(si_c, s), \sigma' \rangle \rightarrow \langle \ll Si^{\rightarrow}_{si,}, G_\pi>, si'>, \sigma' \rangle \ \langle ASK\_BEHAVIOR(si, g), \sigma' \rangle \rightarrow \langle findSitnPolcyRltns(si), \sigma' \rangle \ \langle addBehavior(b^{\pi}_{si}), \sigma' \rangle \rightarrow \langle \overrightarrow{B^{\pi}_{si}}, \sigma' \rangle}{\sigma' \xrightarrow{return\left(\overrightarrow{B^{\pi}_{si}}\right)} \sigma} \ si' = si$$ *return summarized behavior if the last processed situation is the* ***closest*** *to the initial* si_c

Ask_Rule_Behavior (conditions) – Ask for a rule-based behavior

75) $$\frac{\sigma \xrightarrow{ask\_rule\_behavior(iff)} \sigma'}{\langle ASK\_RULE\_BEHAVIOR(iff),\sigma'\rangle \rightarrow \langle searchRulesConditions(iff),\sigma'\rangle}$$

$searchRulesConditions(iff)$ *finds an ordered set of rules to be fired under the* ***sole fulfillment*** *of the* $\boldsymbol{iff}$ *conditions; the search is to be done in the set of rules where* $\boldsymbol{iff}$ *will be mapped to the rules conditions;*

76) $$\frac{\sigma \xrightarrow{ask\_rule\_behavior(iff)} \sigma' \ \langle searchRulesConditions(iff),\sigma'\rangle \rightarrow \langle R_{iff},\sigma'\rangle}{\forall r \in R_{iff} \bullet \langle recordBehavior(r.DO\_ACTIONS),\sigma'\rangle \rightarrow \langle b^{r}_{iff},\sigma'\rangle}$$

KnowLang Grammar*:*
Rule-Body-Decl | IF Bln-Expr THEN Do-Section
Rule-Body-Decl | IF Bln-Expr THEN Do-Section ELSE Do-Section
Rule-Body-Decl | IF Bln-Expr THEN Do-Section ELSE Rule-Body-Decl
Do-Section | Do-Actions
Do-Section | Do-Predicates
Do-Actions | DO_ACTIONS { Action-Names }

77) $$\frac{\sigma \xrightarrow{ask\_rule\_behavior(iff)} \sigma' \ \langle searchRulesConditions(iff),\sigma'\rangle \rightarrow \langle R_{iff},\sigma'\rangle \ \langle recordBehavior(r.DO\_ACTIONS),\sigma'\rangle \rightarrow \langle b^{r}_{iff},\sigma'\rangle}{\langle addBehavior\left(b^{r}_{iff}\right),\sigma'\rangle \rightarrow \langle \overrightarrow{B^{r}_{iff}},\sigma'\rangle}$$

78) $$\frac{\sigma \xrightarrow{ask\_rule\_behavior(iff)} \sigma' \ \langle searchRulesConditions(iff),\sigma'\rangle \rightarrow \langle R_{iff},\sigma'\rangle \ \langle addBehavior\left(b^{r}_{iff}\right),\sigma'\rangle \rightarrow \langle \overrightarrow{B^{r}_{iff}},\sigma'\rangle \ \langle lastRule(R_{iff},r),\sigma'\rangle \rightarrow \langle TRUE,\sigma'\rangle}{\sigma' \xrightarrow{return\left(\overrightarrow{B^{r}_{iff}}\right)} \sigma}$$

Ask_Curr_State (object) – Ask for current state of an object

Note*: The state evaluation will work only if there is a predefined predicate that evaluates the state of the desired object.*

79) $$\frac{\sigma \xrightarrow{ask\_curr\_state(s)} \sigma'}{\langle ASK\_CURR\_STATE(s), \sigma' \rangle \rightarrow \langle findKRObject(s), \sigma' \rangle}$$

s*-implementation reference string; pass the* ***environment object*** *implementation string if you want to get the environment state;*

KnowLang Grammar*:*
Object-Impl | IMPL { Impl-Reference }

$findKRObject(s)$ *maps a SO object to KR object by using the implementation reference string declared at the IMPL section of the object KR;*

80) $$\frac{\sigma \xrightarrow{ask\_curr\_state(s)} \sigma' \ \langle findKRObject(s), \sigma' \rangle \rightarrow \langle o, \sigma' \rangle}{\langle getObjectConcept(o), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}$$

81) $$\frac{\sigma \xrightarrow{ask\_curr\_state(s)} \sigma' \ \langle getObjectConcept(o), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}{\langle getStates(c), \sigma' \rangle \rightarrow \langle c.STATES, \sigma' \rangle}$$

KnowLang Grammar*:*
Concept-States | STATES { Concept-States-Seqnce }
Concept-States | FINAL STATES { Concept-States-Seqnce }
Concept-States-Seqnce | Concept-State Concept-States-Seqnce
Concept-State | STATE State-Name { State-Body }
Concept-State | FINAL STATE State-Name { State-Body }
Concept-State | STATE State-Name { }
Concept-State | FINAL STATE State-Name { }
State-Name | stateIdentifier
State-Body | Bln-Expr

82) $$\frac{\sigma \xrightarrow{ask\_curr\_state(s)} \sigma' \ \langle getObjectConcept(o), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle \ \langle getStates(c), \sigma' \rangle \rightarrow \langle c.STATES, \sigma' \rangle}{\langle evaluatePredicates(o,c,c.STATES), \sigma' \rangle \rightarrow \langle o.STATE, \sigma' \rangle}$$

83) $$\frac{\sigma \xrightarrow{ask\_curr\_state(s)} \sigma' \ \langle evaluatePredicates(o,c,c.STATES), \sigma' \rangle \rightarrow \langle o.STATE, \sigma' \rangle}{\sigma' \xrightarrow{return(o.STATE)} \sigma}$$

Ask_Curr_State – Ask for current system state

84) $$\frac{\sigma \xrightarrow{ask\_curr\_state()} \sigma'}{\langle ASK\_CURR\_STATE(), \sigma' \rangle \rightarrow \langle getSystemConcept(), \sigma' \rangle}$$

85) $$\frac{\sigma \xrightarrow{ask\_curr\_state(s)} \sigma' \ \langle getSystemConcept(), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle}{\langle getStates(c), \sigma' \rangle \rightarrow \langle c.STATES, \sigma' \rangle}$$

86) $$\frac{\sigma \xrightarrow{ask\_curr\_state()} \sigma' \ \langle getSystemConcept(), \sigma' \rangle \rightarrow \langle c, \sigma' \rangle \ \langle getStates(c), \sigma' \rangle \rightarrow \langle c.STATES, \sigma' \rangle}{\langle evaluateSystemPredicates(c.STATES), \sigma' \rangle \rightarrow \langle this.STATE, \sigma' \rangle}$$

87) $$\frac{\sigma \xrightarrow{ask\_curr\_state()} \sigma' \ \langle evaluateSystemPredicates(c.STATES), \sigma' \rangle \rightarrow \langle this.STATE, \sigma' \rangle}{\sigma' \xrightarrow{return(this.STATE)} \sigma}$$

Ask_Curr_Situation – Ask for current situation

88) $$\frac{\sigma \xrightarrow{ask\_curr\_situation()} \sigma'}{\langle ASK\_CURR\_SITUATION(), \sigma' \rangle \rightarrow \langle findCurrentSituation(), \sigma' \rangle}$$

89) $$\frac{\sigma \xrightarrow{ask\_curr\_situation()} \sigma' \ \langle findCurrentSituation(), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}{\sigma' \xrightarrow{return(si)} \sigma}$$

90) $$\frac{\langle findCurrentSituation(), \sigma' \rangle \rightarrow \langle NIL, \sigma' \rangle \ \langle ASK\_CURR\_STATE(), \sigma' \rangle \rightarrow \langle s, \sigma' \rangle}{\langle findClosestSituation\ (s), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}$$

$findClosestSituation(s)$ *finds a situation that has a state "closest" to the current state* ***s***;

91) $$\frac{\sigma \xrightarrow{ask\_curr\_situation()} \sigma' \ \langle findClosestSituation\ (s), \sigma' \rangle \rightarrow \langle si, \sigma' \rangle}{\sigma' \xrightarrow{return(si)} \sigma}$$