

Grammar for the English to LTL Parser

August 2009

There are four types of sentences that can be written in this grammar: initial conditions, environment assumptions, motion/action requirements and conditional sentences. None of these sentences is required for the simulation - the parser has a default for every part of the LTL formula φ (see Section ??).

All of these sentences must be constructed using the following templates and the list of valid propositions. For this grammar we define:

Valid propositions := sensor propositions, region names, robot actions and auxiliary propositions defined by the user.

ϕ := Boolean Formula composed from the valid propositions and boolean connectives (parenthesis may be added)

$$\phi ::= \text{a valid proposition} \mid \text{not } \phi \mid \phi \text{ or } \phi \mid \phi \text{ and } \phi \mid \phi \text{ implies } \phi \mid \phi \text{ iff } \phi$$

ϕ_{env} := Boolean formula composed **only from valid sensor propositions** and boolean connectives (parenthesis may also be added)

ϕ_{region} := Boolean formula composed **only from valid region names** and boolean connectives (parenthesis may also be added)

ϕ_{action} := Boolean formula composed **only from robot actions and auxiliary propositions** and boolean connectives (parenthesis may also be added)

ϕ_{robot} := Boolean formula composed from all valid propositions **except the sensor propositions** and boolean connectives (parenthesis may also be added)

Note that the following templates, except for the propositions, are *not* case sensitive. Furthermore, any line starting with # is considered a comment and is ignored.

1 Initial Conditions

Valid initial condition sentences for the environment are:

- “Environment starts with ϕ_{env} ” - ϕ_{env} is the initial condition for the sensor propositions
- “Environment starts with true” - all sensor propositions are initially set to true
- “Environment starts with false” - all sensor propositions are initially set to false

Valid initial condition sentences for the robot are:

- “Robot starts in ϕ_{region} ” - ϕ_{region} sets the possible initial regions for the robot. If this sentence is omitted, the robot can start in any defined region.
- “Robot starts with ϕ_{action} ” - ϕ_{action} sets the initial values of the robot actions and the auxiliary propositions. Propositions that are not part of ϕ_{action} can be initially either true or false.
- “Robot starts with true” - all robot actions and auxiliary propositions are initially set to true
- “Robot starts with false” - all robot actions and auxiliary propositions are initially set to false

- “Robot starts in ϕ_{region} with ϕ_{action} ”
- “Robot starts in ϕ_{region} with true”
- “Robot starts in ϕ_{region} with false”

Note: Environment may be replaced with “Env” and “Robot starts” may be replaced with “You start”.

2 Environment Assumptions and Motion/Action Requirements

We distinguish between two types of assumptions/requirements: **Safety** and **Liveness**. In the following, for environment assumptions $\varphi = \phi_{env}$ and for robot requirements $\varphi = \phi_{robot}$. This means that the sensor and robot propositions **cannot** be mixed in these basic instructions.

2.1 Safety

Safety assumptions and requirements specify behaviors that must always be met, for example (in natural language): “Never enter the haunted room with a cake”.

Valid and equivalent safety sentences are:

- “Always φ ”
- “Always do φ ”
- “Do φ ”

2.2 Liveness

Liveness assumptions and requirements specify goals that always must eventually be met, thus describing situations that must be reached infinitely often. For example (in natural language) “Go to the haunted room infinitely often”.

Valid and equivalent liveness sentences are:

- “Go to φ ”
- “Visit φ ”
- “Infinitely often do φ ”
- “Infinitely often φ ”

2.3 Go To and Stay

We allow one more type of robot requirement, **Go To and Stay**:

- “Go to ϕ_{robot} and stay there”
- “Go to ϕ_{robot} and stay”

While we allow ϕ_{robot} , the intended use of this sentence is for statements such as “Go to the haunted room and stay there”.

3 Conditionals

There are three types of conditional sentences that are allowed:

- “If *condition* then *requirement*”
- “*requirement* unless *condition*”
- “*requirement* if and only if *condition*”

3.1 Requirements

A requirement is either a **safety** or a **liveness** sentence. As such, it should contain either sensor propositions or robot propositions, but not both.

A requirement can also be “**Stay there**” or “**Stay**”.

3.2 Conditions

Conditions are built up by combining basic conditions using *and* and *or* (which are also case insensitive). The structure of a basic condition is “*Phrase ϕ* ”, where *Phrase* is one of the following:

	Positive	Negative
Past	‘The robot was in’ ‘You were in’ ‘It was in’ ‘Were in’ ‘Was in’ ‘The robot sensed’ ‘You sensed’ ‘It sensed’ ‘Sensed’ ‘The robot activated’ ‘You activated’ ‘It activated’ ‘Activated’	‘The robot was not in’ ‘You were not in’ ‘It was not in’ ‘Were not in’ ‘Was not in’ ‘The robot did not sense’ ‘You did not sense’ ‘It did not sense’ ‘Did not sense’ ‘The robot did not activate’ ‘You did not activate’ ‘It did not activate’ ‘Did not activate’
Present	‘The robot is in’ ‘You are in’ ‘It is in’ ‘Are in’ ‘Is in’ ‘The robot is sensing’ ‘You are sensing’ ‘It is sensing’ ‘Are sensing’ ‘Is sensing’ ‘The robot is activating’ ‘You are activating’ ‘It is activating’ ‘Are activating’ ‘Is activating’	‘The robot is not in’ ‘You are not in’ ‘It is not in’ ‘Are not in’ ‘Is not in’ ‘The robot is not sensing’ ‘You are not sensing’ ‘It is not sensing’ ‘Are not sensing’ ‘Is not sensing’ ‘The robot is not activating’ ‘You are not activating’ ‘It is not activating’ ‘Are not activating’ ‘Is not activating’

A few notes about the tenses:

Tenses are used to decide whether the ‘next’ operator needs to be attached to a proposition. Due to the structure of the LTL formulas we consider, the ‘next’ operator is *not* used in liveness requirements, therefore, both the past and present tenses are the same for such propositions.

For safety requirements (including ‘stay’), the past tense relates to formulas that are true in the current state while the present tense relates to formulas that should be true in the next state.

For environment safety (again, due to the structure of the formula and the nature of the synthesis algorithm) all robot propositions *must* be in the past tense. If they are not, the parser will translate them into the past tense and issue a warning.

One should keep in mind that the past tense only relates to formulas (situations) that are true in the current state. If the desired behavior needs to depend on events that took place in the past, an auxiliary proposition should be created to ‘remember’ that the event took place. See Section 4 for some macros.

4 Macros

Macros are defined to provide a succinct way to specify some useful notions.

4.1 Memory Propositions

As mentioned in Section 3.2, if the specification depends on events that happened in the past, and not just in the previous state, an auxiliary proposition needs to be defined and used as memory.

To assist with that we define the macro:

- “*MemProp* is set on ϕ_{set} and reset on ϕ_{reset} ”
- “*MemProp* is set on ϕ_{set} and reset on false”

where *MemProp* is the auxiliary proposition (which is a robot proposition) and ϕ_{set} and ϕ_{reset} are boolean formulas in the set of all propositions.

MemProp becomes true whenever ϕ_{set} is true *and* ϕ_{reset} is false. This means that if both formulas are true, *MemProp* will remain false. Once it is true, the proposition remains true while ϕ_{reset} is false and once it is false it remains false while ϕ_{set} is false. In other words, the proposition is true from the time (state) ϕ_{set} is true until, but not including, the time (state) in which ϕ_{reset} is true.

In the case of “*MemProp* is set on ϕ_{set} and reset on false”, once *MemProp* becomes true it stays true forever.

As an example, if the desired behavior is to beep from the time the robot sensed Mika in Room 3 until it senses her mother, the specification would look like this:

Propositions: { Mika, Mother, Room3, Beep, NeedToBeep }

Spec:

Describing the behavior of the auxiliary proposition NeedToBeep:

NeedToBeep is set on Room3 and Mika and reset on Mother

Describe the beeping:

Beep if and only if you are activating NeedToBeep