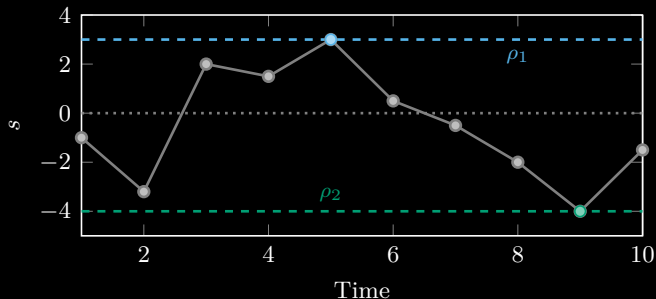


# SignalTemporalLogic.jl

## INTRODUCTION

$$\psi_1 = \Diamond(s_t > 0) \quad \psi_2 = \Box(s_t > 0)$$



ROBERT MOSS

STANFORD AA228V/CS238V

mossr@cs.stanford.edu

# INSTALLATION

# INSTALLATION

You can install the `SignalTemporalLogic.jl` package via:

```
using Pkg  
Pkg.add("SignalTemporalLogic")
```

# INSTALLATION

You can install the `SignalTemporalLogic.jl` package via:

```
using Pkg  
Pkg.add("SignalTemporalLogic")
```

Then you can run this to use the package:

```
using SignalTemporalLogic
```

# SPECIFICATIONS

Let's define the following *specification* over a trajectory  $\tau$ :

# SPECIFICATIONS

Let's define the following *specification* over a trajectory  $\tau$ :

$$\psi(\tau) = \Diamond(s_t > 0) \quad \text{“Eventually } (\Diamond), \text{ the state will be greater than zero.”}$$

# SPECIFICATIONS

Let's define the following *specification* over a trajectory  $\tau$ :

$$\psi(\tau) = \Diamond(s_t > 0) \quad \text{“Eventually } (\Diamond), \text{ the state will be greater than zero.”}$$

```
julia> using SignalTemporalLogic
```

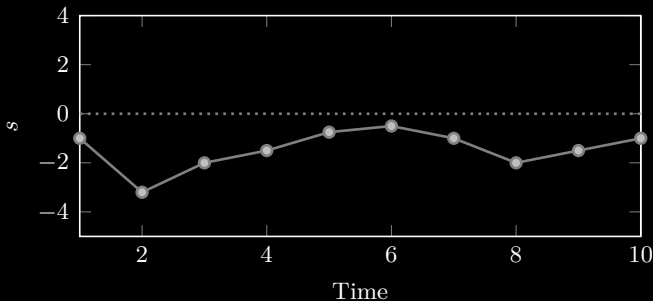
# SPECIFICATIONS

Let's define the following *specification* over a trajectory  $\tau$ :

$$\psi(\tau) = \Diamond(s_t > 0) \quad \text{“Eventually } (\Diamond), \text{ the state will be greater than zero.”}$$

```
julia> using SignalTemporalLogic
```

```
julia>  $\tau$  = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
```





# SPECIFICATIONS

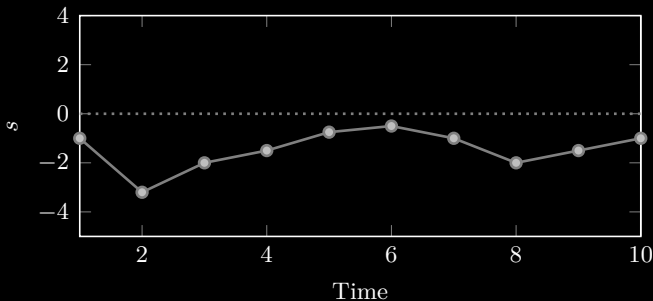
Let's define the following *specification* over a trajectory  $\tau$ :

$$\psi(\tau) = \Diamond(s_t > 0) \quad \text{“Eventually } (\Diamond), \text{ the state will be greater than zero.”}$$

```
julia> using SignalTemporalLogic
```

```
julia>  $\tau$  = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
```

```
 $\tau$  # \tau<TAB>
```



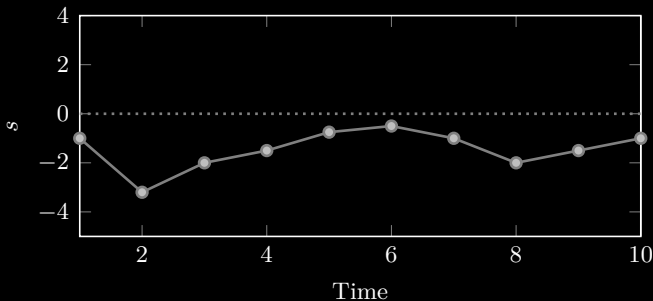
# SPECIFICATIONS

Let's define the following *specification* over a trajectory  $\tau$ :

$$\psi(\tau) = \Diamond(s_t > 0) \quad \text{“Eventually } (\Diamond), \text{ the state will be greater than zero.”}$$

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
julia>  $\psi$  = @formula  $\Diamond(s_t \rightarrow s_t > 0)$ ;
```

```
 $\tau$  # \tau<TAB>
```



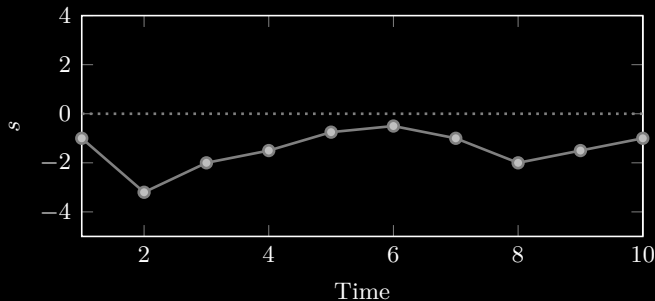
# SPECIFICATIONS

Let's define the following *specification* over a trajectory  $\tau$ :

$$\psi(\tau) = \Diamond(s_t > 0) \quad \text{“Eventually } (\Diamond), \text{ the state will be greater than zero.”}$$

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
julia>  $\psi$  = @formula  $\Diamond(s_t \rightarrow s_t > 0)$ ;
```

```
 $\tau$  # \tau<TAB>
 $\psi$  # \psi<TAB>
 $\Diamond$  # \lozenge<TAB>
```



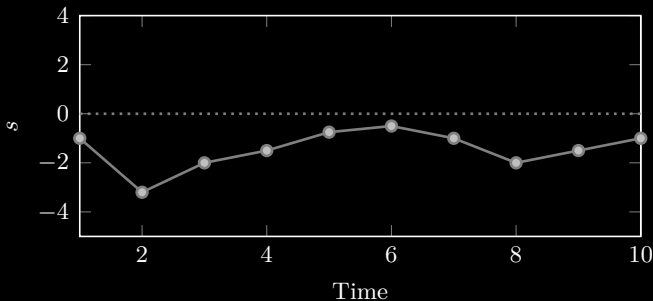
# SPECIFICATIONS

Let's define the following *specification* over a trajectory  $\tau$ :

$$\psi(\tau) = \Diamond(s_t > 0) \quad \text{“Eventually } (\Diamond), \text{ the state will be greater than zero.”}$$

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
julia>  $\psi$  = @formula  $\Diamond(s_t \rightarrow s_t > 0)$ ;
julia>  $\psi(\tau)$ 
false
```

```
 $\tau$  # \tau<TAB>
 $\psi$  # \psi<TAB>
 $\Diamond$  # \lozenge<TAB>
```



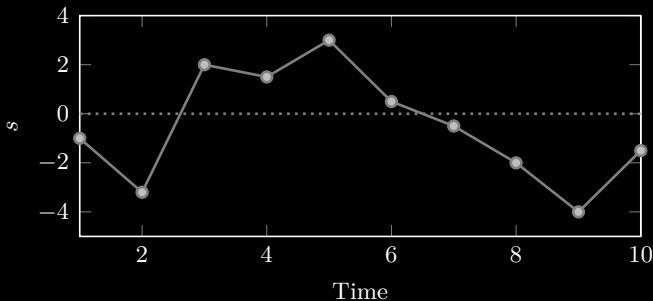
# SPECIFICATIONS

Let's define the following *specification* over a trajectory  $\tau$ :

$$\psi(\tau) = \Diamond(s_t > 0) \quad \text{“Eventually } (\Diamond), \text{ the state will be greater than zero.”}$$

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia>  $\psi$  = @formula  $\Diamond(s_t \rightarrow s_t > 0)$ ;
julia>  $\psi(\tau)$ 
true
```

```
 $\tau$  # \tau<TAB>
 $\psi$  # \psi<TAB>
 $\Diamond$  # \lozenge<TAB>
```



# ROBUSTNESS

You can also compute the *robustness* of a trajectory  $\tau$ .

# ROBUSTNESS

You can also compute the *robustness* of a trajectory  $\tau$ .

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia>  $\psi_1$  = @formula  $\diamond(s_t \rightarrow s_t > 0)$ ;
```

# ROBUSTNESS

You can also compute the *robustness* of a trajectory  $\tau$ .

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia>  $\psi_1$  = @formula  $\diamond(s_t \rightarrow s_t > 0)$ ;
julia>  $\rho_1$  =  $\rho(\tau, \psi_1)$ 
3.0
```

```
 $\rho$  # \rho<TAB>
```



# ROBUSTNESS

You can also compute the *robustness* of a trajectory  $\tau$ .

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia>  $\psi_1$  = @formula  $\diamond(s_t \rightarrow s_t > 0)$ ;
julia>  $\rho_1$  =  $\rho(\tau, \psi_1)$ 
3.0
julia>  $\psi_2$  = @formula  $\square(s_t \rightarrow s_t > 0)$ ;
```

$\rho$  # \rho<TAB>

$\square$  # \square<TAB>

# ROBUSTNESS

You can also compute the *robustness* of a trajectory  $\tau$ .

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia>  $\psi_1$  = @formula  $\diamond(s_t \rightarrow s_t > 0)$ ;
julia>  $\rho_1$  =  $\rho(\tau, \psi_1)$ 
3.0
julia>  $\psi_2$  = @formula  $\square(s_t \rightarrow s_t > 0)$ ;
julia>  $\rho_2$  =  $\rho(\tau, \psi_2)$ 
-4.0
```

```
 $\rho$  # \rho<TAB>
```

```
 $\square$  # \square<TAB>
```

# ROBUSTNESS

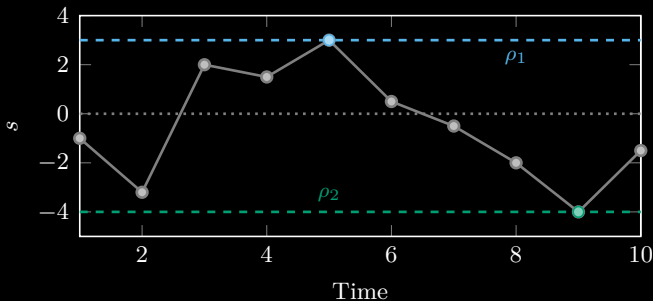
You can also compute the *robustness* of a trajectory  $\tau$ .

```
julia> using SignalTemporalLogic
julia>  $\tau$  = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia>  $\psi_1$  = @formula  $\Diamond(s_t \rightarrow s_t > 0)$ ;
julia>  $\rho_1$  =  $\rho(\tau, \psi_1)$ 
3.0
julia>  $\psi_2$  = @formula  $\Box(s_t \rightarrow s_t > 0)$ ;
julia>  $\rho_2$  =  $\rho(\tau, \psi_2)$ 
-4.0
```

$\rho$  # \rho<TAB>

$\Box$  # \square<TAB>

$$\psi_1 = \Diamond(s_t > 0) \quad \psi_2 = \Box(s_t > 0)$$



# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

*First, let's define the `Eventually` operator in Julia.*

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

*First, let's define the `Eventually` operator in Julia.*

```
mutable struct Eventually <: Formula
     $\psi$ ::Formula
    I::Interval
end
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

*First, let's define the `Eventually` operator in Julia.*

```
mutable struct Eventually <: Formula
     $\psi$ ::Formula
    I::Interval
end
```

$$\Diamond_{[a,b]}\psi = \top \mathcal{U}_{[a,b]}\psi$$

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

*First, let's define the `Eventually` operator in Julia.*

```
mutable struct Eventually <: Formula
     $\psi$ ::Formula
    I::Interval
end
```

$$\begin{aligned}\Diamond_{[a,b]}\psi &= \top \mathcal{U}_{[a,b]}\psi \\ &= \exists t(a \leq t \leq b)\psi_t\end{aligned}$$

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

*First, let's define the `Eventually` operator in Julia.*

```
mutable struct Eventually <: Formula
     $\psi$ ::Formula
    I::Interval
end
```

$$\begin{aligned}\Diamond_{[a,b]}\psi &= \top \mathcal{U}_{[a,b]}\psi \\ &= \exists t(a \leq t \leq b)\psi_t\end{aligned}$$

```
( $\Diamond$ ::Eventually)(x) = any( $\Diamond$ . $\psi$ (x[t]) for t in interval( $\Diamond$ ,x)) # Evaluate formula  $\psi$ (x)
```



# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

$$\rho(x_t, \diamond_{[a,b]}\psi) = \max_{t' \in [t+a, t+b]} \rho(x_{t'}, \psi)$$

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

$$\rho(x_t, \Diamond_{[a,b]}\psi) = \max_{t' \in [t+a, t+b]} \rho(x_{t'}, \psi)$$

```
 $\rho(x, \Diamond::\text{Eventually}) = \text{maximum}(\rho(x[t], \Diamond.\psi) \text{ for } t \in \text{interval}(\Diamond, x)) \text{ \# Robustness}$ 
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

$$\rho(x_t, \Diamond_{[a,b]}\psi) = \max_{t' \in [t+a, t+b]} \rho(x_{t'}, \psi)$$

```
ρ(x, ◇::Eventually) = maximum(ρ(x[t], ◇.ψ) for t ∈ interval(◇, x)) # Robustness
```

$$\tilde{\rho}(x_t, \Diamond_{[a,b]}\psi) = \widetilde{\max}_{t' \in [t+a, t+b]} \tilde{\rho}(x_{t'}, \psi)$$

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

$$\rho(x_t, \Diamond_{[a,b]}\psi) = \max_{t' \in [t+a, t+b]} \rho(x_{t'}, \psi)$$

```
ρ(x, ◇::Eventually) = maximum(ρ(x[t], ◇.ψ) for t ∈ interval(◇,x)) # Robustness
```

$$\tilde{\rho}(x_t, \Diamond_{[a,b]}\psi) = \widetilde{\max}_{t' \in [t+a, t+b]} \tilde{\rho}(x_{t'}, \psi)$$

```
̃ρ(x, ◇::Eventually; w=1) = smoothmax(̃ρ(x[t], ◇.ψ; w) for t ∈ interval(◇,x); w) # Smooth robustness
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

$$\rho(x_t, \Diamond_{[a,b]}\psi) = \max_{t' \in [t+a, t+b]} \rho(x_{t'}, \psi)$$

```
ρ(x, ◇::Eventually) = maximum(ρ(x[t], ◇.ψ) for t ∈ interval(◇,x)) # Robustness
```

$$\tilde{\rho}(x_t, \Diamond_{[a,b]}\psi) = \widetilde{\max}_{t' \in [t+a, t+b]} \tilde{\rho}(x_{t'}, \psi)$$

```
̃ρ(x, ◇::Eventually; w=1) = smoothmax(̃ρ(x[t], ◇.ψ; w) for t ∈ interval(◇,x); w) # Smooth robustness
```

$$\widetilde{\max}(x; w) = \frac{\sum_i^n x_i \exp(x_i/w)}{\sum_j^n \exp(x_j/w)}$$

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

$$\rho(x_t, \Diamond_{[a,b]}\psi) = \max_{t' \in [t+a, t+b]} \rho(x_{t'}, \psi)$$

```
 $\rho(x, \Diamond::\text{Eventually}) = \text{maximum}(\rho(x[t], \Diamond.\psi) \text{ for } t \in \text{interval}(\Diamond, x)) \text{ \# Robustness}$ 
```

$$\tilde{\rho}(x_t, \Diamond_{[a,b]}\psi) = \widetilde{\max}_{t' \in [t+a, t+b]} \tilde{\rho}(x_{t'}, \psi)$$

```
 $\tilde{\rho}(x, \Diamond::\text{Eventually}; w=1) = \text{smoothmax}(\tilde{\rho}(x[t], \Diamond.\psi; w) \text{ for } t \in \text{interval}(\Diamond, x); w) \text{ \# Smooth robustness}$ 
```

$$\widetilde{\max}(x; w) = \frac{\sum_i^n x_i \exp(x_i/w)}{\sum_j^n \exp(x_j/w)}$$

```
 $\text{smoothmax}(x; w=1) = (w == 0) ? \text{maximum}(x) : \text{sum}(x_i * \exp(x_i/w) \text{ for } x_i \text{ in } x) / \text{sum}(\exp(x_j/w) \text{ for } x_j \text{ in } x)$ 
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

*This means we can use Julia's **auto-differentiation** packages right out-of-the-box!*



# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

*This means we can use Julia's **auto-differentiation** packages right out-of-the-box!*

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia>  $\tau = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5]$ ;
julia>  $\psi = \text{@formula } \Diamond(s_t \rightarrow s_t > 0)$ ;
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\Diamond$

*This means we can use Julia's **auto-differentiation** packages right out-of-the-box!*

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia>  $\tau = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5]$ ;
julia>  $\psi = \text{@formula } \phi(s_t \rightarrow s_t > 0)$ ;
julia> gradient( $\tau \rightarrow \rho(\tau, \psi)$ ,  $\tau$ )
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

*This means we can use Julia's **auto-differentiation** packages right out-of-the-box!*

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia>  $\tau = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5]$ ;
julia>  $\psi = @formula \diamond(s_t \rightarrow s_t > 0)$ ;
julia> gradient( $\tau \rightarrow \rho(\tau, \psi)$ ,  $\tau$ )
10-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 1.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

*This means we can use Julia's *auto-differentiation* packages right out-of-the-box!*

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia>  $\tau = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];$ 
julia>  $\psi = @formula \diamond(s_t \rightarrow s_t > 0);$ 
julia> gradient( $\tau \rightarrow \rho(\tau, \psi), \tau$ )
10-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 1.0
 0.0
 0.0
 0.0
 0.0
 0.0

# Smooth robustness gradient
julia> gradient( $\tau \rightarrow \tilde{\rho}(\tau, \psi), \tau$ )
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

*This means we can use Julia's **auto-differentiation** packages right out-of-the-box!*

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia>  $\tau = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];$ 
julia>  $\psi = @formula \diamond(s_t \rightarrow s_t > 0);$ 
julia> gradient( $\tau \rightarrow \rho(\tau, \psi)$ ,  $\tau$ )
# Smooth robustness gradient
julia> gradient( $\tau \rightarrow \tilde{\rho}(\tau, \psi)$ ,  $\tau$ )
10-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 1.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

$\tilde{\rho} \# \backslash \rho < \text{TAB} > \backslash \tilde{} < \text{TAB} >$

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

*This means we can use Julia's **auto-differentiation** packages right out-of-the-box!*

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia>  $\tau$  = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia>  $\psi$  = @formula  $\diamond(s_t \rightarrow s_t > 0)$ ;
julia> gradient( $\tau \rightarrow \rho(\tau, \psi)$ ,  $\tau$ )
10-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 1.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

```
# Smooth robustness gradient
julia> gradient( $\tau \rightarrow \tilde{\rho}(\tau, \psi)$ ,  $\tau$ )
10-element Vector{Float64}:
-0.02435977847707571
-0.0052615588731724375
 0.14412350994359485
 0.023385630608315888
 0.9656907041997265
-0.038507325550849236
-0.03149700775134967
-0.012828520312110245
-0.002782850376909859
-0.01796280341016957
```

```
 $\tilde{\rho}$  # \rho<TAB>\tilde<TAB>
```

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

*The textbook (and projects) use aliases for  $\rho$  and  $\tilde{\rho}$*

# ROBUSTNESS AND SMOOTH ROBUSTNESS

EXAMPLE: EVENTUALLY  $\diamond$

*The textbook (and projects) use aliases for  $\rho$  and  $\tilde{\rho}$*

`robustness(s,  $\psi$ ) =  $\rho$ (s,  $\psi$ )`

`smooth_robustness(s,  $\psi$ ; w=1) =  $\tilde{\rho}$ (s,  $\psi$ ; w=1)`



## USE IN PROJECTS

Wrappers are provided in the textbook/projects:

# USE IN PROJECTS

Wrappers are provided in the textbook/projects:

*Linear temporal logic (LTL)*

```
struct LTLSpecification <: Specification
    formula # formula specified using SignalTemporalLogic.jl
end
evaluate( $\psi$ ::LTLSpecification,  $\tau$ ) =  $\psi$ .formula([step.s for step in  $\tau$ ])
```

# USE IN PROJECTS

Wrappers are provided in the textbook/projects:

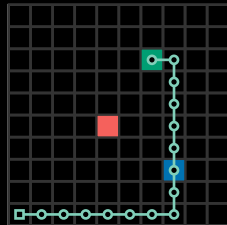
*Linear temporal logic (LTL)*

```
struct LTLSpecification <: Specification
    formula # formula specified using SignalTemporalLogic.jl
end
evaluate( $\psi$ ::LTLSpecification,  $\tau$ ) =  $\psi$ .formula([step.s for step in  $\tau$ ])
```

*Signal temporal logic (STL, includes time interval)*

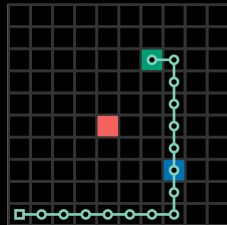
```
struct STLSpecification <: Specification
    formula # formula specified using SignalTemporalLogic.jl
    I       # time interval (e.g. 3:10)
end
evaluate( $\psi$ ::STLSpecification,  $\tau$ ) =  $\psi$ .formula([step.s for step in  $\tau$ [ $\psi$ .I]])
```

# GRID WORLD



# GRID WORLD

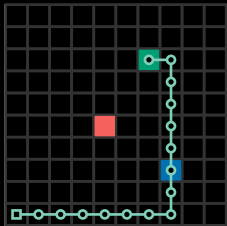
$F(s_t)$  : the state  $s$  at time  $t$  contains an obstacle



# GRID WORLD

$F(s_t)$  : the state  $s$  at time  $t$  contains an obstacle

$G(s_t)$  : the state  $s$  at time  $t$  is the goal

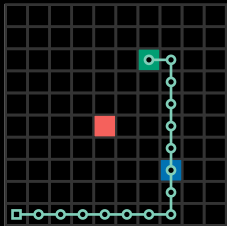


# GRID WORLD

$F(s_t)$  : the state  $s$  at time  $t$  contains an obstacle

$G(s_t)$  : the state  $s$  at time  $t$  is the goal

$C(s_t)$  : the state  $s$  at time  $t$  is the checkpoint



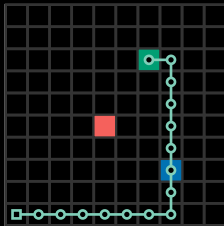
# GRID WORLD

$F(s_t)$  : the state  $s$  at time  $t$  contains an obstacle

$G(s_t)$  : the state  $s$  at time  $t$  is the goal

$C(s_t)$  : the state  $s$  at time  $t$  is the checkpoint

$$\psi = \underbrace{\Diamond G}_{\text{reaches goal}}$$





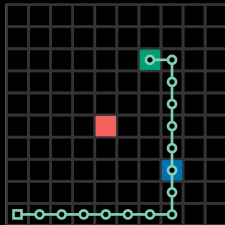
# GRID WORLD

$F(s_t)$  : the state  $s$  at time  $t$  contains an obstacle

$G(s_t)$  : the state  $s$  at time  $t$  is the goal

$C(s_t)$  : the state  $s$  at time  $t$  is the checkpoint

$$\psi = \underbrace{\Diamond G}_{\text{reaches goal}} \wedge \underbrace{\neg GUC}_{\text{reach checkpoint before goal}}$$



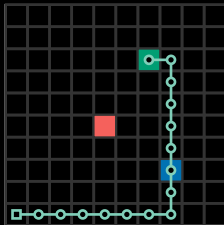
# GRID WORLD

$F(s_t)$  : the state  $s$  at time  $t$  contains an obstacle

$G(s_t)$  : the state  $s$  at time  $t$  is the goal

$C(s_t)$  : the state  $s$  at time  $t$  is the checkpoint

$$\psi = \underbrace{\Diamond G}_{\text{reaches goal}} \wedge \underbrace{\neg GUC}_{\text{reach checkpoint before goal}} \wedge \underbrace{\Box \neg F}_{\text{always avoid obstacles}}$$



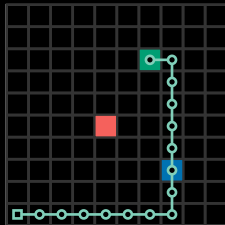
# GRID WORLD

$F(s_t)$  : the state  $s$  at time  $t$  contains an obstacle

$G(s_t)$  : the state  $s$  at time  $t$  is the goal

$C(s_t)$  : the state  $s$  at time  $t$  is the checkpoint

$$\psi = \underbrace{\Diamond G}_{\text{reaches goal}} \wedge \underbrace{\neg GUC}_{\text{reach checkpoint before goal}} \wedge \underbrace{\Box \neg F}_{\text{always avoid obstacles}}$$



```
F = @formula s_t -> s_t == [5, 5]
G = @formula s_t -> s_t == [7, 8]
C = @formula s_t -> s_t == [8, 3]
ψ = LTLSpecification(@formula ◇(G) ∧ U(¬G, C) ∧ □(¬F))
```


# CONTINUUM WORLD

System	Property	Implementation


# CONTINUUM WORLD

System	Property	Implementation
Continuum World		
		

# CONTINUUM WORLD

System	Property	Implementation
Continuum World 	<i>“Reach the goal without hitting the obstacle”</i> $G(s_t)$ : $s_t$ is in the goal region $F(s_t)$ : $s_t$ is in the obstacle region $\psi = \Diamond G \wedge \Box \neg F$	

# CONTINUUM WORLD

System	Property	Implementation
Continuum World 	<i>“Reach the goal without hitting the obstacle”</i> $G(s_t)$ : $s_t$ is in the goal region $F(s_t)$ : $s_t$ is in the obstacle region $\psi = \Diamond G \wedge \Box \neg F$	$G = \text{@formula } s \rightarrow \text{norm}(s - [6.5, 7.5]) \leq 0.5$ $F = \text{@formula } s \rightarrow \text{norm}(s - [4.5, 4.5]) \leq 0.5$ $\psi = \text{@formula } \Diamond(G) \wedge \Box(\neg F)$

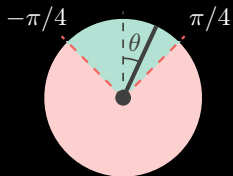
# INVERTED PENDULUM

System	Property	Implementation

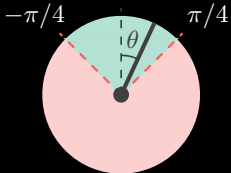


# INVERTED PENDULUM

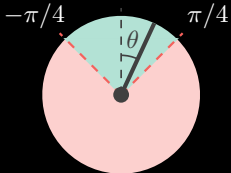
System	Property	Implementation
Inverted Pendulum		



# INVERTED PENDULUM

System	Property	Implementation
Inverted Pendulum 	<i>“Keep the pendulum balanced”</i> $B(s_t):  \theta_t  \leq \pi/4$ $\psi = \Box B$	

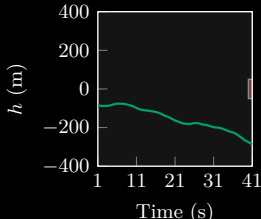
# INVERTED PENDULUM

System	Property	Implementation
<p>Inverted Pendulum</p> 	<p><i>“Keep the pendulum balanced”</i></p> $B(s_t):  \theta_t  \leq \pi/4$ $\psi = \Box B$	$B = \text{@formula } s \rightarrow \text{abs}(s[1]) \leq \pi/4$ $\psi = \text{@formula } \Box(B)$

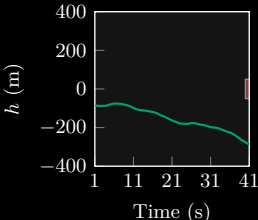
# AIRCRAFT COLLISION AVOIDANCE

System	Property	Implementation

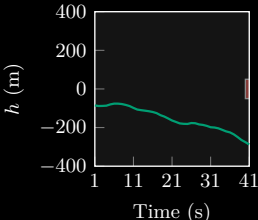
# AIRCRAFT COLLISION AVOIDANCE

System	Property	Implementation
Aircraft Collision Avoidance		
		

# AIRCRAFT COLLISION AVOIDANCE

System	Property	Implementation
<p>Aircraft Collision Avoidance</p>  <p>Time (s)</p>	<p><i>“Ensure at least 50 meters relative altitude between 40 and 41 seconds”</i></p> <p><math>S(s_t):  h_t  \geq 50</math></p> <p><math>\psi = \Box_{[40,41]} S</math></p>	

# AIRCRAFT COLLISION AVOIDANCE

System	Property	Implementation
<p>Aircraft Collision Avoidance</p>  <p><math>h</math> (m)</p> <p>Time (s)</p>	<p><i>“Ensure at least 50 meters relative altitude between 40 and 41 seconds”</i></p> <p><math>S(s_t):  h_t  \geq 50</math></p> <p><math>\psi = \Box_{[40,41]} S</math></p>	<p><math>S = \text{@formula } s \rightarrow \text{abs}(s[1]) \geq 50</math></p> <p><math>\psi = \text{@formula } \Box(40:41, S)</math></p>

# RESOURCES



# RESOURCES



`github.com/sisl/SignalTemporalLogic.jl`

# RESOURCES



[github.com/sisl/SignalTemporalLogic.jl](https://github.com/sisl/SignalTemporalLogic.jl)



[github.com/mossr/STL-mini-lecture](https://github.com/mossr/STL-mini-lecture)