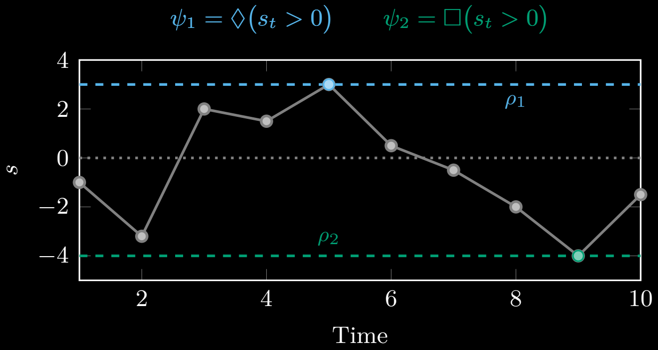
SignalTemporalLogic.jl

Introduction



Robert Moss
Stanford AA228V/CS238V
mossr@cs.stanford.edu

Installation

Installation

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

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

Installation

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

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

Then you can run this to use the package:

```julia
using SignalTemporalLogic
```

Specifications

Let's define the following *specification* over a trajectory τ:

Specifications

Let's define the following *specification* over a trajectory τ:

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

Specifications

Let's define the following *specification* over a trajectory τ:

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

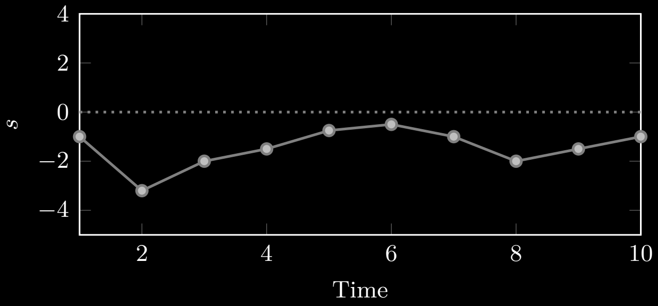
```
julia> using SignalTemporalLogic
```

Specifications

Let's define the following *specification* over a trajectory τ:

$$\psi(\tau) = \lozenge\big(s_t > 0\big) \qquad \textit{“Eventually } (\lozenge)\textit{, the state will be greater than zero.”}$$

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
```



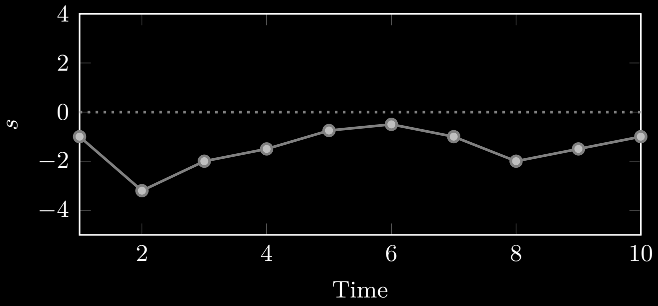
Specifications

Let's define the following *specification* over a trajectory τ:

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

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
```

```
τ # \tau<TAB>
```



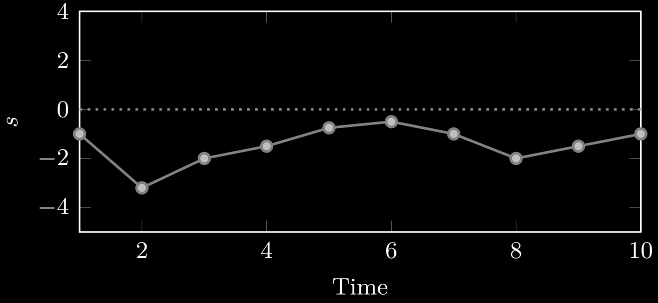
Specifications

Let's define the following *specification* over a trajectory τ:

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

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
julia> ψ = @formula ◊(sₜ -> sₜ > 0);
```

```
τ # \tau<TAB>
```



Specifications

Let's define the following *specification* over a trajectory τ:

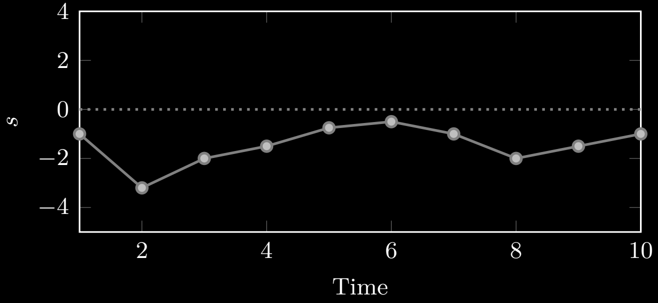
$$\psi(\tau) = \lozenge(s_t > 0) \qquad \text{“\textit{Eventually} } (\lozenge)\text{\textit{, the state will be greater than zero.}”}$$

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
julia> ψ = @formula ◊(sₜ -> sₜ > 0);
```

```
τ # \tau<TAB>

ψ # \psi<TAB>

◊ # \lozenge<TAB>
```



Specifications

Let's define the following *specification* over a trajectory τ:

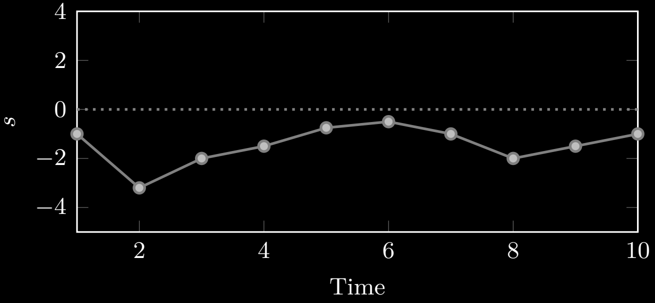
$$\psi(\tau) = \lozenge(s_t > 0) \qquad \text{“\textit{Eventually} } (\lozenge)\text{\textit{, the state will be greater than zero.}”}$$

```julia
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, -2.0, -1.5, -0.75, -0.5, -1.0, -2.0, -1.5, -1.0];
julia> ψ = @formula ◊(sₜ -> sₜ > 0);
julia> ψ(τ)
false
```

```
τ # \tau<TAB>

ψ # \psi<TAB>

◊ # \lozenge<TAB>
```



Specifications

Let's define the following *specification* over a trajectory τ:

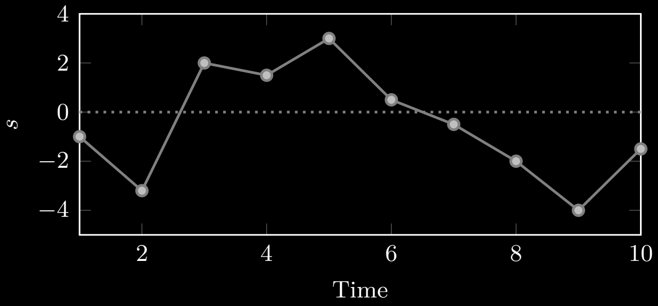
$$\psi(\tau) = \Diamond(s_t > 0) \qquad \text{“\textit{Eventually} } (\Diamond)\text{\textit{, the state will be greater than zero.}”}$$

```julia
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ = @formula ◊(sₜ -> sₜ > 0);
julia> ψ(τ)
true
```

```
τ # \tau<TAB>

ψ # \psi<TAB>

◊ # \lozenge<TAB>
```



Robustness

You can also compute the *robustness* of a trajectory τ.

Robustness

You can also compute the *robustness* of a trajectory τ.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ → sₜ > 0);
```

Robustness

You can also compute the *robustness* of a trajectory τ.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ → sₜ > 0);
julia> ρ₁ = ρ(τ, ψ₁)
3.0
```

```
ρ # \rho<TAB>
```

Robustness

You can also compute the *robustness* of a trajectory τ.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ → sₜ > 0);
julia> ρ₁ = ρ(τ, ψ₁)
3.0
julia> ψ₂ = @formula □(sₜ → sₜ > 0);
```

```
ρ # \rho<TAB>
□ # \square<TAB>
```

Robustness

You can also compute the *robustness* of a trajectory τ.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ -> sₜ > 0);
julia> ρ₁ = ρ(τ, ψ₁)
3.0
julia> ψ₂ = @formula □(sₜ -> sₜ > 0);
julia> ρ₂ = ρ(τ, ψ₂)
-4.0
```

```
ρ # \rho<TAB>
□ # \square<TAB>
```

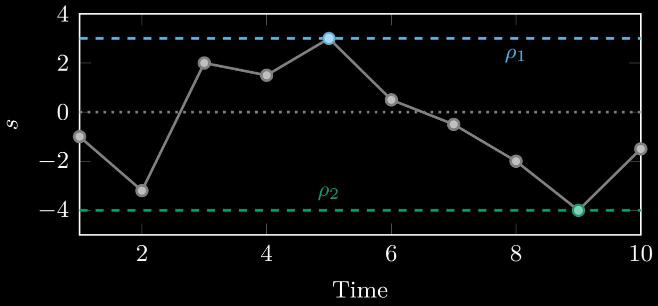
Robustness

You can also compute the *robustness* of a trajectory τ.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ -> sₜ > 0);
julia> ρ₁ = ρ(τ, ψ₁)
3.0
julia> ψ₂ = @formula □(sₜ -> sₜ > 0);
julia> ρ₂ = ρ(τ, ψ₂)
-4.0
```

```
ρ # \rho<TAB>
□ # \square<TAB>
```

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



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.*

```julia
mutable struct Eventually <: Formula
    ψ::Formula
    I::Interval
end
```

Robustness and Smooth Robustness

Example: Eventually $\Diamond$

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

```
mutable struct Eventually <: Formula
    ψ::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
    ψ::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 $\lozenge$

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

```julia
mutable struct Eventually <: Formula
    ψ::Formula
    I::Interval
end
```

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

```julia
(◊::Eventually)(x) = any(◊.ψ(x[t]) for t ∈ interval(◊,x)) # Evaluate formula ψ(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 $\lozenge$

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

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

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
```

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

```
ρ̃(x, ◊::Eventually; w=1) = smoothmax(ρ̃(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
```

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

```
ρ̃(x, ◊::Eventually; w=1) = smoothmax(ρ̃(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 $\Diamond$

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

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

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

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

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

```
smoothmax(x; w=1) = (w == 0) ? maximum(x) : sum(xᵢ*exp(xᵢ/w) for xᵢ in x) / sum(exp(xⱼ/w) for xⱼ 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 ◊

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

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ = @formula ◊(sₜ → sₜ > 0);
```

Robustness and Smooth Robustness

Example: Eventually $\lozenge$

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

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ = @formula ◊(sₜ → sₜ > 0);
julia> gradient(τ→ρ(τ, ψ), τ)
```

Robustness and Smooth Robustness

Example: Eventually $\lozenge$

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

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ = @formula ◊(sₜ → sₜ > 0);
julia> gradient(τ→ρ(τ, ψ), τ)
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 ◊

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

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ = @formula ◊(sₜ → sₜ > 0);
julia> gradient(τ->ρ(τ, ψ), τ)
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(τ->ρ̃(τ, ψ), τ)
```

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> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ = @formula ◊(sₜ -> sₜ > 0);
julia> gradient(τ->ρ(τ, ψ), τ)
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(τ->ρ̃(τ, ψ), τ)
```

```
ρ̃ # \rho<TAB>\tilde<TAB>
```

Robustness and Smooth Robustness

Example: Eventually ◊

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

```
julia> using SignalTemporalLogic
julia> import ForwardDiff: gradient
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ = @formula ◊(sₜ → sₜ > 0);
julia> gradient(τ→ρ(τ, ψ), τ)
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(τ→ρ̃(τ, ψ), τ)
10-element Vector{Float64}:
 -0.024359778477075 71
 -0.0052615588731724375
  0.14412350994359485
  0.023385630608315888
  0.9656907041997265
 -0.038507325550849236
 -0.03149700775134967
 -0.012828520312110245
 -0.002782850376909859
 -0.01796280341016957
```

```
ρ̃ # \rho<TAB>\tilde<TAB>
```

Robustness and Smooth Robustness

Example: Eventually $\Diamond$

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

Robustness and Smooth Robustness

Example: Eventually $\Diamond$

The textbook (and projects) use aliases for ρ *and* ρ̃

```
robustness(s, ψ) = ρ(s, ψ)
smooth_robustness(s, ψ; w=1) = ρ̃(s, ψ; 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(ψ::LTLSpecification, τ) = ψ.formula([step.s for step in τ])
```

Use in Projects

Wrappers are provided in the textbook/projects:

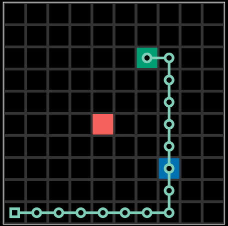
Linear temporal logic (LTL)

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

Signal temporal logic (STL, includes time interval)

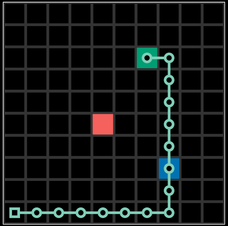
```
struct STLSpecification <: Specification
    formula # formula specified using SignalTemporalLogic.jl
    I       # time interval (e.g. 3:10)
end
evaluate(ψ::STLSpecification, τ) = ψ.formula([step.s for step in τ[ψ.I]])
```

Grid World



Grid World

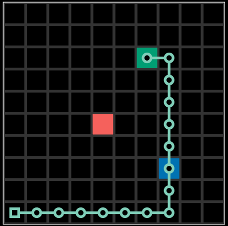
$F(s_t)$: the state s at time t contains an obstacle



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

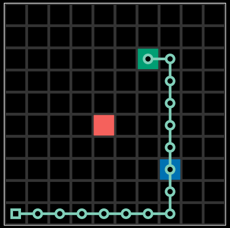


Grid World

$F(s_t)$: 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



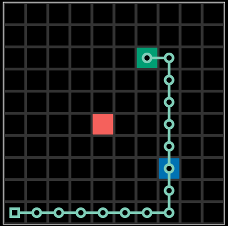
Grid World

$F(s_t)$: 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

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



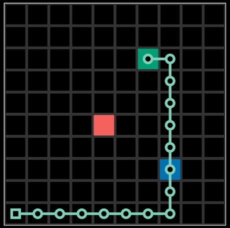
Grid World

$F(s_t)$: 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

$$\psi = \underbrace{\Diamond G(s_t)}_{\text{reaches goal}} \wedge \underbrace{\neg C(s_t)\,\mathcal{U}\,G(s_t)}_{\text{reach checkpoint before goal}}$$



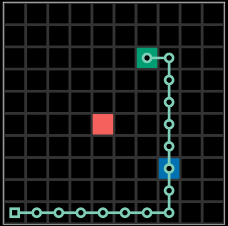
Grid World

$F(s_t)$: 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

$$\psi = \underbrace{\Diamond G(s_t)}_{\text{reaches goal}} \wedge \underbrace{\neg C(s_t)\, \mathcal{U}\, G(s_t)}_{\text{reach checkpoint before goal}} \wedge \underbrace{\Box \neg F(s_t)}_{\text{always avoid obstacles}}$$



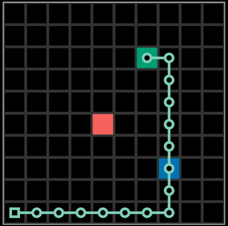
Grid World

$F(s_t)$: 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

$$\psi = \underbrace{\Diamond G(s_t)}_{\text{reaches goal}} \wedge \underbrace{\neg C(s_t) \,\mathcal{U}\, G(s_t)}_{\text{reach checkpoint before goal}} \wedge \underbrace{\square \neg F(s_t)}_{\text{always avoid obstacles}}$$

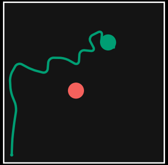


```
F = @formula sₜ -> sₜ == [5, 5]
G = @formula sₜ -> sₜ == [7, 8]
C = @formula sₜ -> sₜ == [8, 3]
ψ = LTLSpecification(@formula ◊(G) ∧ 𝒰(¬G, C) ∧ □(¬F))
```

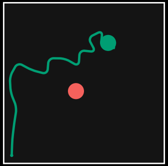
Continuum World

System	Property	Implementation

Continuum World

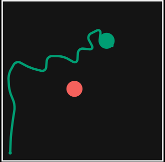
System	Property	Implementation
Continuum World 		

Continuum World

System	Property	Implementation
Continuum World 	*"Reach the goal without hitting the obstacle"* $G(s_t)$: s_t is in the goal region $F(s_t)$: s_t is in the obstacle region $\psi = \Diamond G(s_t) \wedge \Box \neg F(s_t)$	

Continuum World

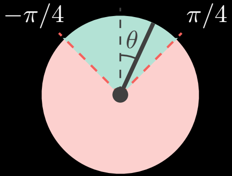
System	Property	Implementation

Continuum World 	*“Reach the goal without hitting the obstacle”*
	$G(s_t)$: s_t is in the goal region
	$F(s_t)$: s_t is in the obstacle region
	$\psi = \Diamond G(s_t) \wedge \Box \neg F(s_t)$
		`G = @formula s→norm(s.-[6.5,7.5])≤0.5`
		`F = @formula s→norm(s.-[4.5,4.5])≤0.5`
		`ψ = @formula ◊(G) ∧ □(¬F)`

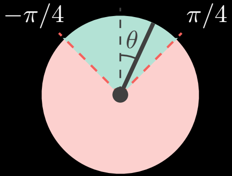
Inverted Pendulum

System	Property	Implementation

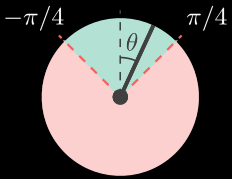
Inverted Pendulum

System	Property	Implementation
Inverted Pendulum 		

Inverted Pendulum

System	Property	Implementation
Inverted Pendulum 	*"Keep the pendulum balanced"* $B(s_t)$: $\lvert\theta_t\rvert \leq \pi/4$ $\psi = \Box B(s_t)$	

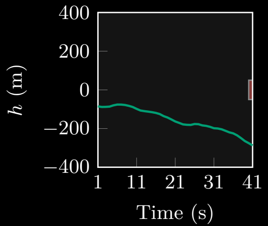
Inverted Pendulum

System	Property	Implementation
Inverted Pendulum 	*"Keep the pendulum balanced"* $B(s_t)$: $\lvert\theta_t\rvert \leq \pi/4$ $\psi = \Box B(s_t)$	`B = @formula s→abs(s[1])≤π/4` `ψ = @formula □(B)`

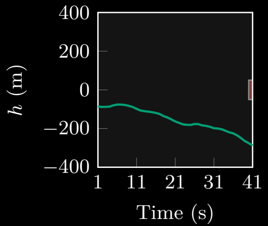
Aircraft Collision Avoidance

System	Property	Implementation

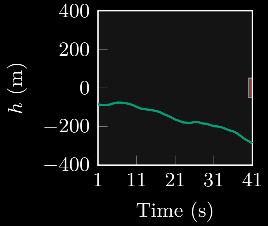
Aircraft Collision Avoidance

System	Property	Implementation
Aircraft Collision Avoidance 		

Aircraft Collision Avoidance

System	Property	Implementation
Aircraft Collision Avoidance 	*"Ensure at least 50 meters relative altitude between 40 and 41 seconds"* $S(s_t)$: $\lvert h_t \rvert \geq 50$ $\psi = \Box_{[40,41]} S(s_t)$	

Aircraft Collision Avoidance

System	Property	Implementation
Aircraft Collision Avoidance 	*"Ensure at least 50 meters relative altitude between 40 and 41 seconds"* $S(s_t)$: $\lvert h_t \rvert \geq 50$ $\psi = \square_{[40,41]} S(s_t)$	`S = @formula s->abs(s[1])≥50` `ψ = @formula □(40:41, S)`

Resources

Resources

github.com/sisl/SignalTemporalLogic.jl

Resources

github.com/sisl/SignalTemporalLogic.jl

github.com/mossr/STL-mini-lecture