Systems Design Laboratory

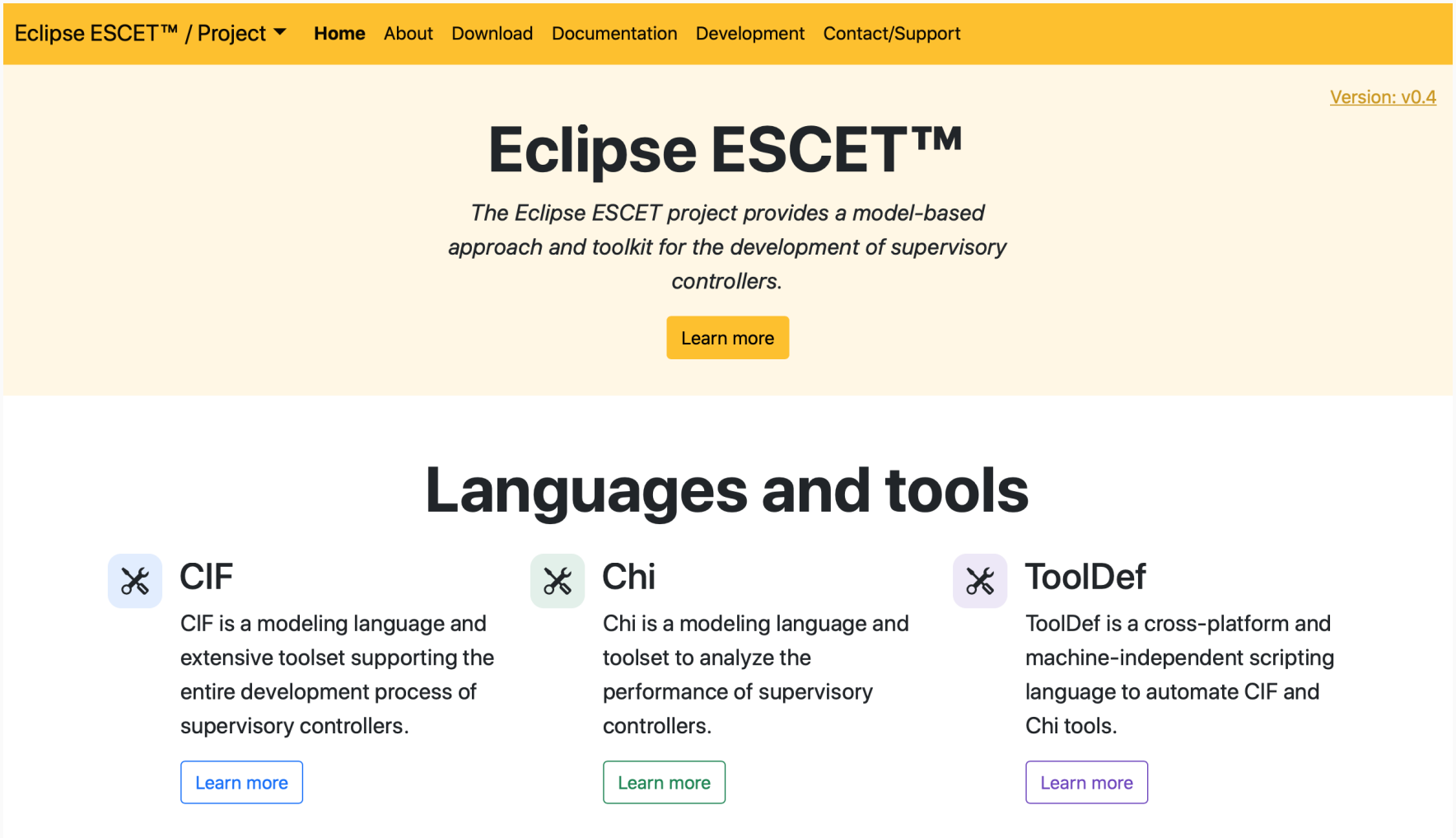
ToolDef

Matteo Zavatteri

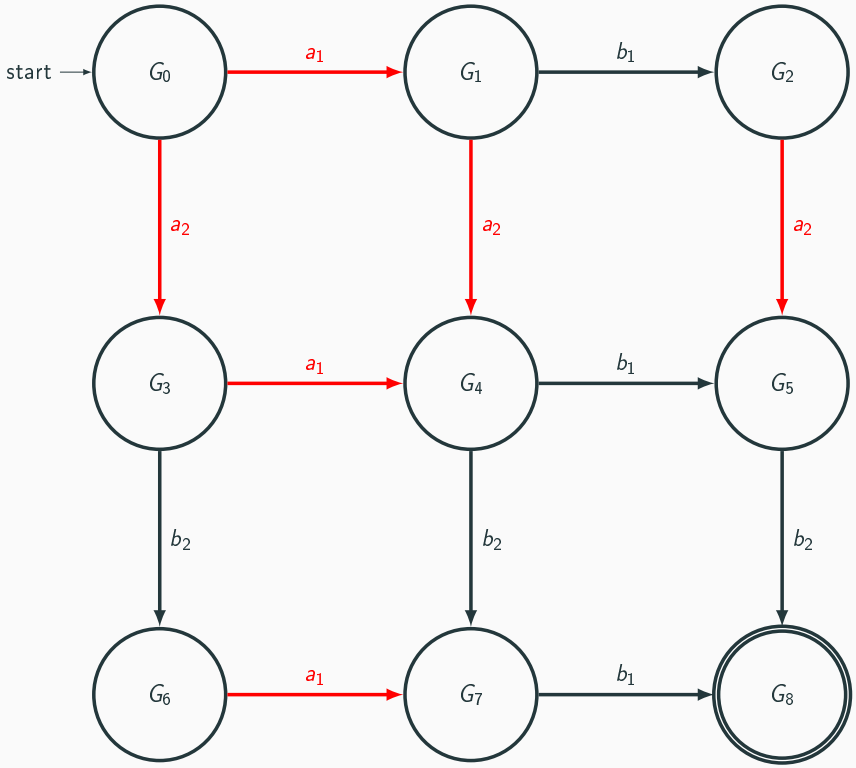
[1]Department of Mathematics, University of Padova, ITALY

[2]Department of Computer Science, University of Verona, ITALY

Recap on the Database Example with CIF and ESCET



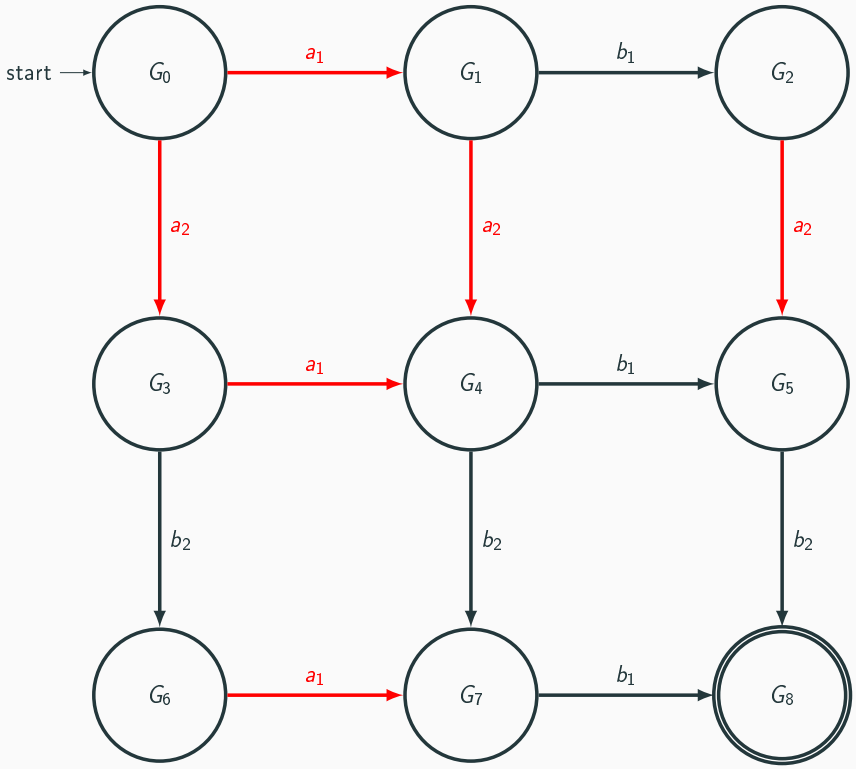
Example 1 - events.cif



- Events a_1, a_2 are uncontrollable
- Events b_1, b_2 are controllable

```
uncontrollable a1, a2;
controllable b1, b2;
```

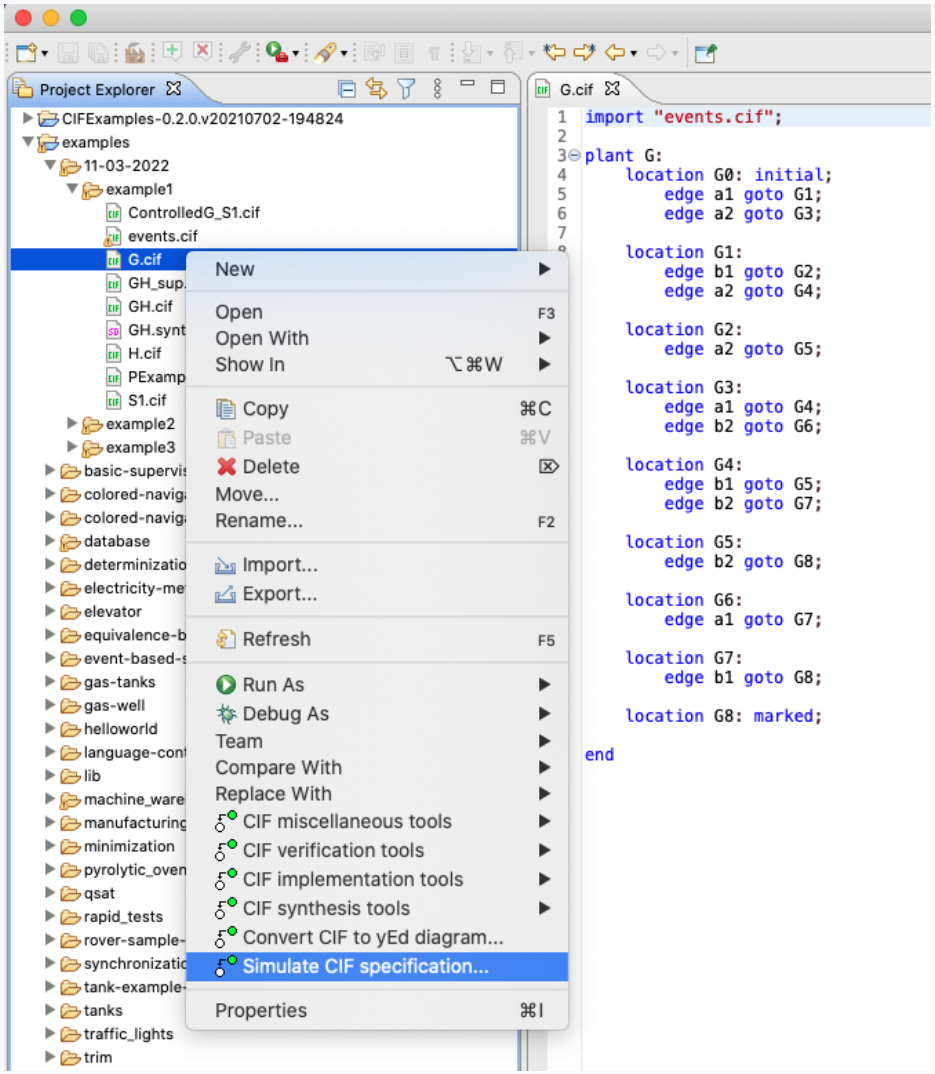
Example 1 - Plant - G.cif



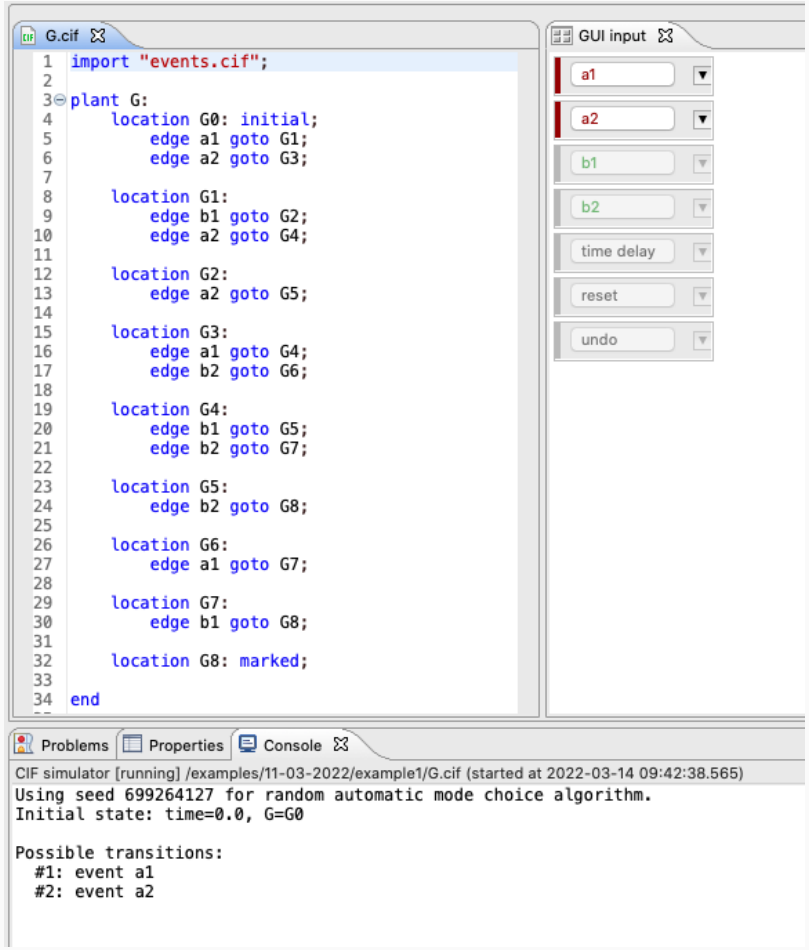

```
import "events.cif";

plant G:
    location G0: initial;
        edge a1 goto G1;
        edge a2 goto G3;

    location G1:
        edge b1 goto G2;
        edge a2 goto G4;

    ...

    location G7:
        edge b1 goto G8;

    location G8: marked;

end
```

Example 1 - Simulation of the Uncontrolled Plant

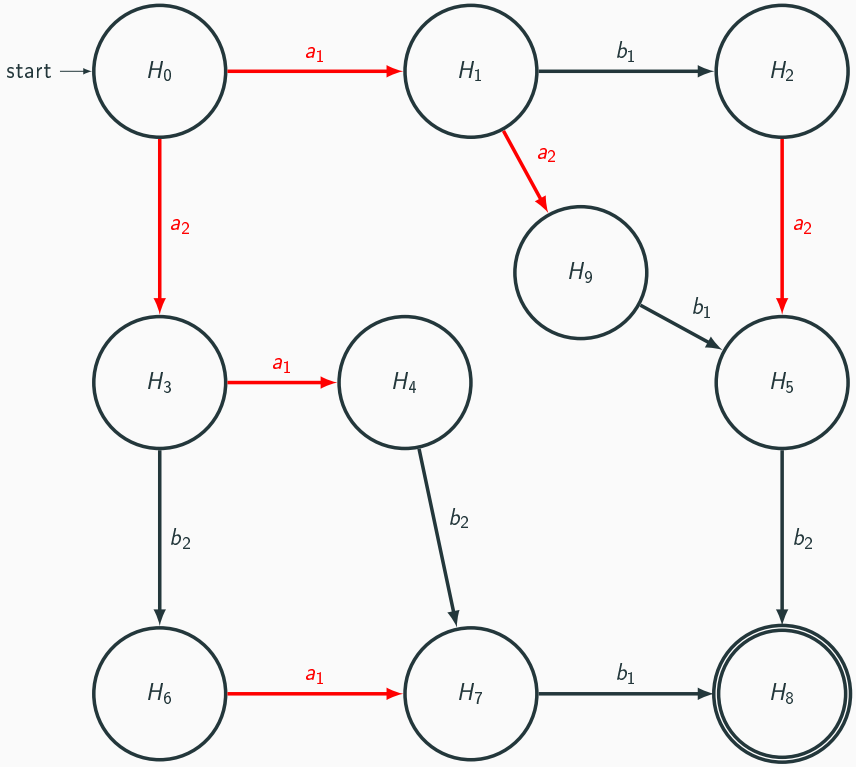
Simulation Menu



Interactive Simulation



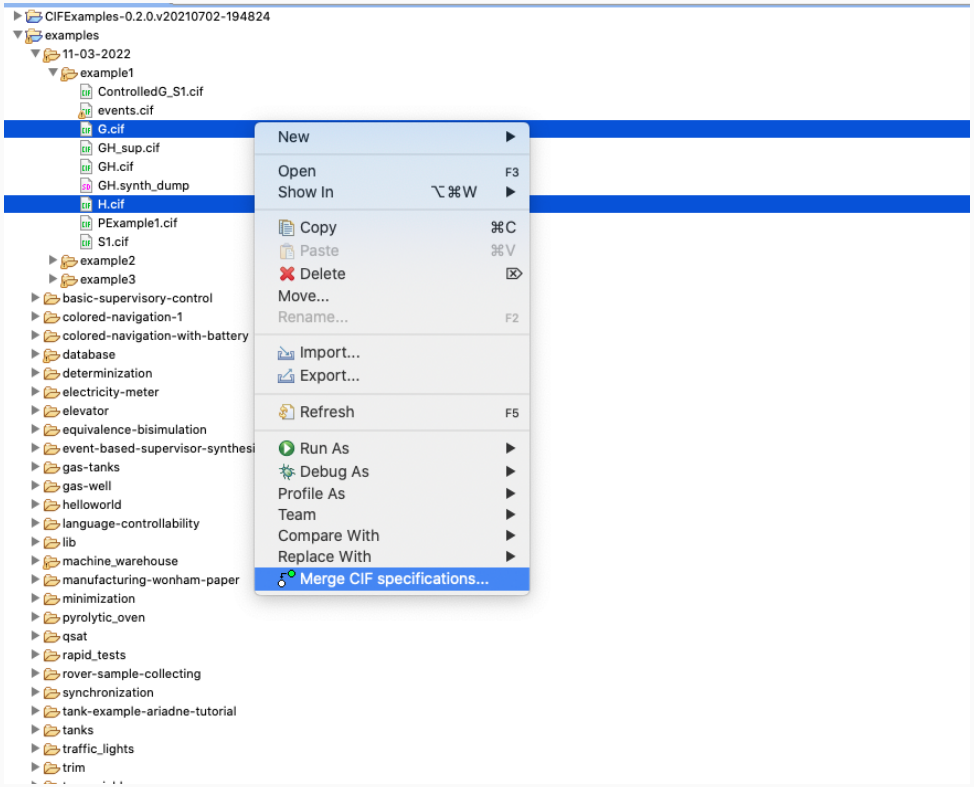
Example 1 - Full Requirement - H.cif



Requirement: a_1 precedes a_2 if and only if b_1 precedes b_2

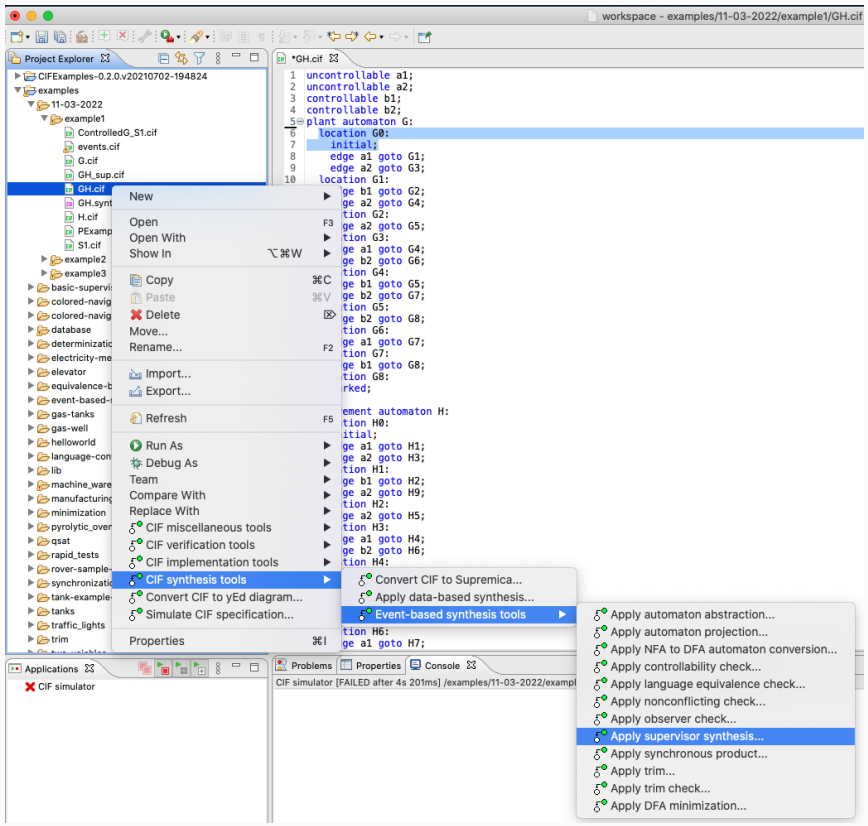
```
import "events.cif";

requirement H:
    location H0: initial;
        edge a1 goto H1;
        edge a2 goto H3;

    ...

    location H4:
        edge b2 goto H7;

    location H9:
        edge b1 goto H5;

    ...

    location H8: marked;

end
```

Example 1 - Merge - GH.cif



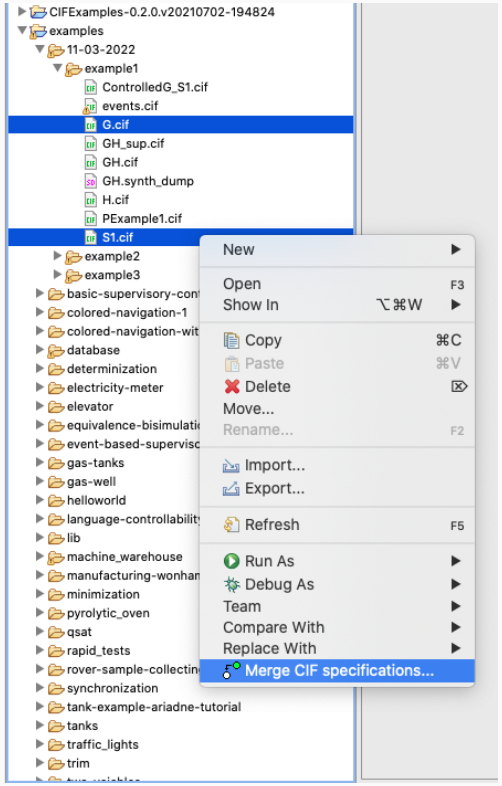

```
uncontrollable a1;
uncontrollable a2;
controllable b1;
controllable b2;

plant automaton G:
  location G0:
    initial;
  ...
  location G8:
    marked;
end

requirement automaton H:
  location H0:
  ...
  location H8:
    marked;
end
```

Example 1 - Supervisor Synthesis - S1.cif




```
uncontrollable a1;
uncontrollable a2;
controllable b1;
controllable b2;

supervisor automaton sup:
  alphabet a1, a2, b1, b2;
  location s1:
    initial;
    edge a2 goto s2;
    edge a1 goto s3;
  location s2:
    edge b2 goto s6;
    edge a1 goto s7;
  ...
  location s9:
    edge b1 goto s10;
  location s10:
    marked;
end
```

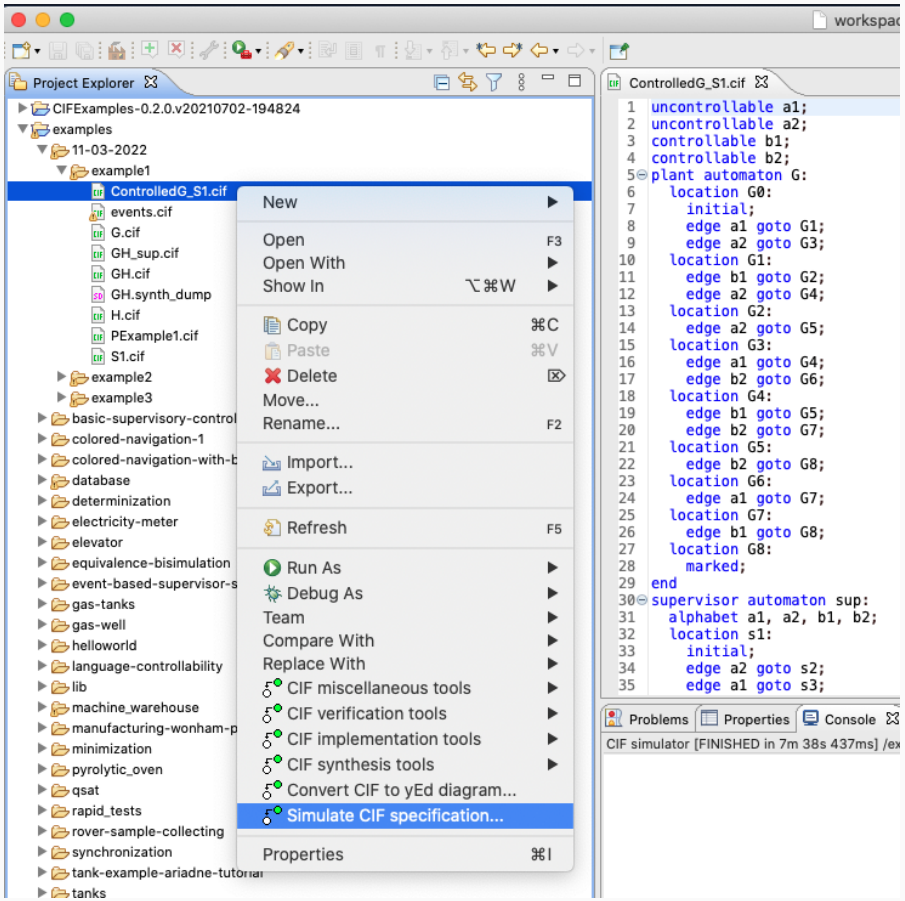
Example 1 - Supervisor Deployment - ControlledG_S1.cif



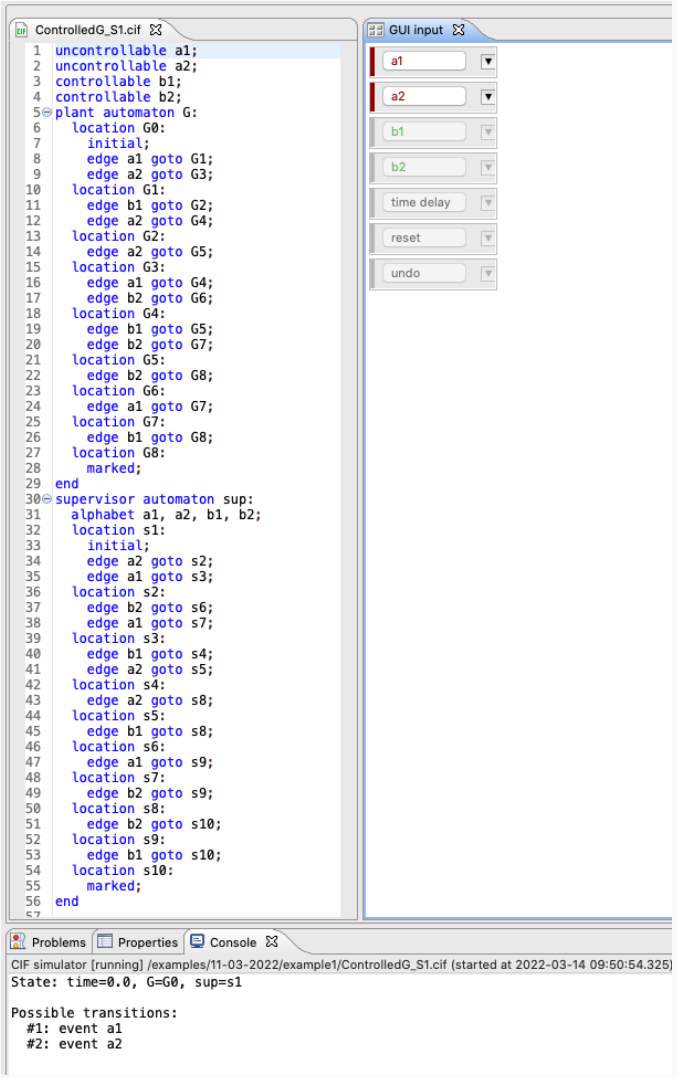

```
uncontrollable a1;
uncontrollable a2;
controllable b1;
controllable b2;

plant automaton G:
  location G0:
    initial;
  ...
  location G8:
    marked;
end

supervisor automaton S1:
  alphabet a1, a2, b1, b2;
  location s1:
    initial;
  ...
  location s10:
    marked;
end
```

Example 1 - Simulation of the Controlled Plant

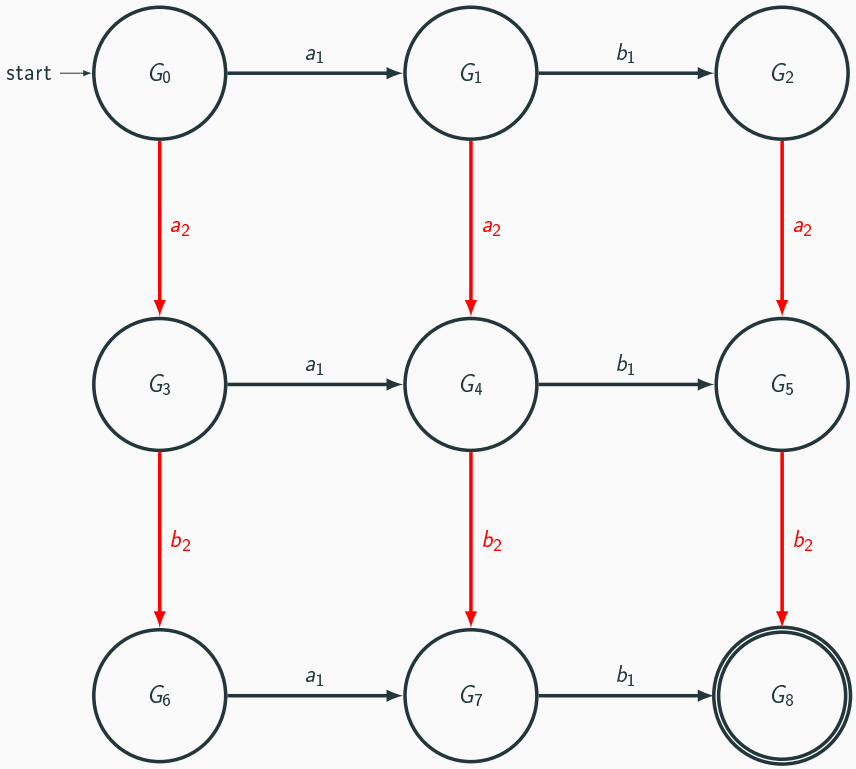
Simulation Menu



Interactive Simulation



Example 2 - Events, Plant, Requirement

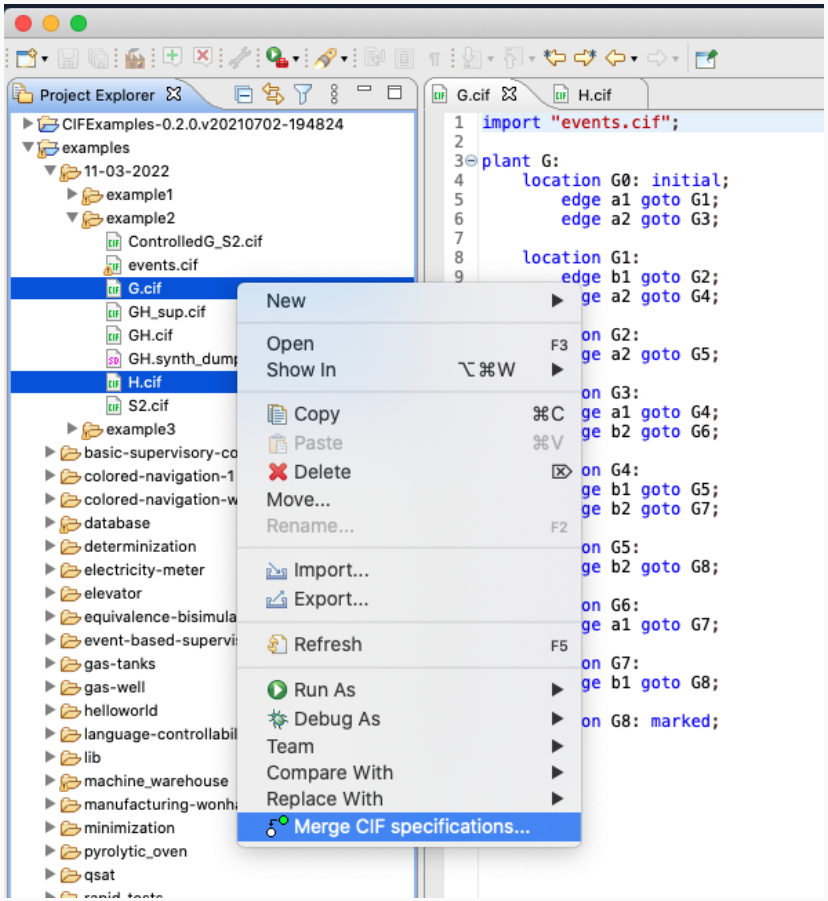


- Events a_1, b_1 are controllable
- Events a_2, b_2 are uncontrollable

```
controllable a1, b1;
uncontrollable a2, b2;
```

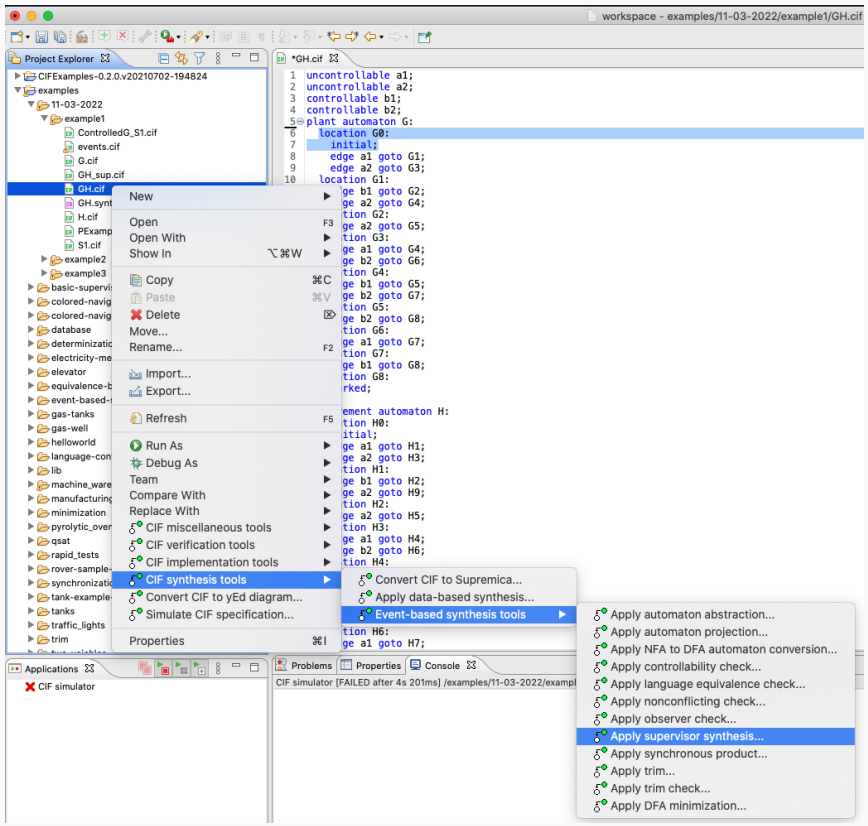
- **We changed only the `event.cif` file**
- **Plant file `G.cif` and Full requirement file `H.cif` are still the same**

Example 2 - Merge - GH.cif



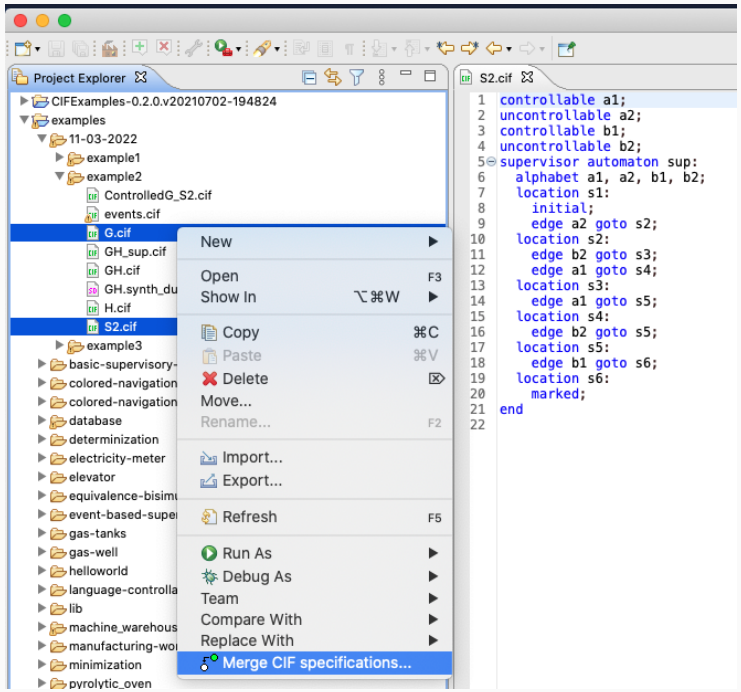

```
controllable a1;
controllable b1;
uncontrollable a2;
uncontrollable b2;

plant automaton G:
  location G0:
    initial;
    ...
  location G8:
    marked;
end

requirement automaton H:
  location H0:
    initial;
    ...
  location H8:
    marked;
end
```

Example 2 - Supervisor Synthesis - S2.cif




```
uncontrollable a1;
uncontrollable a2;
controllable b1;
controllable b2;

supervisor automaton S2:
  alphabet a1, a2, b1, b2;
  location s1:
    initial;
    edge a2 goto s2;
    edge a1 goto s3;
  location s2:
    edge b2 goto s6;
    edge a1 goto s7;
  ...
  location s9:
    edge b1 goto s10;
  location s10:
    marked;
end
```

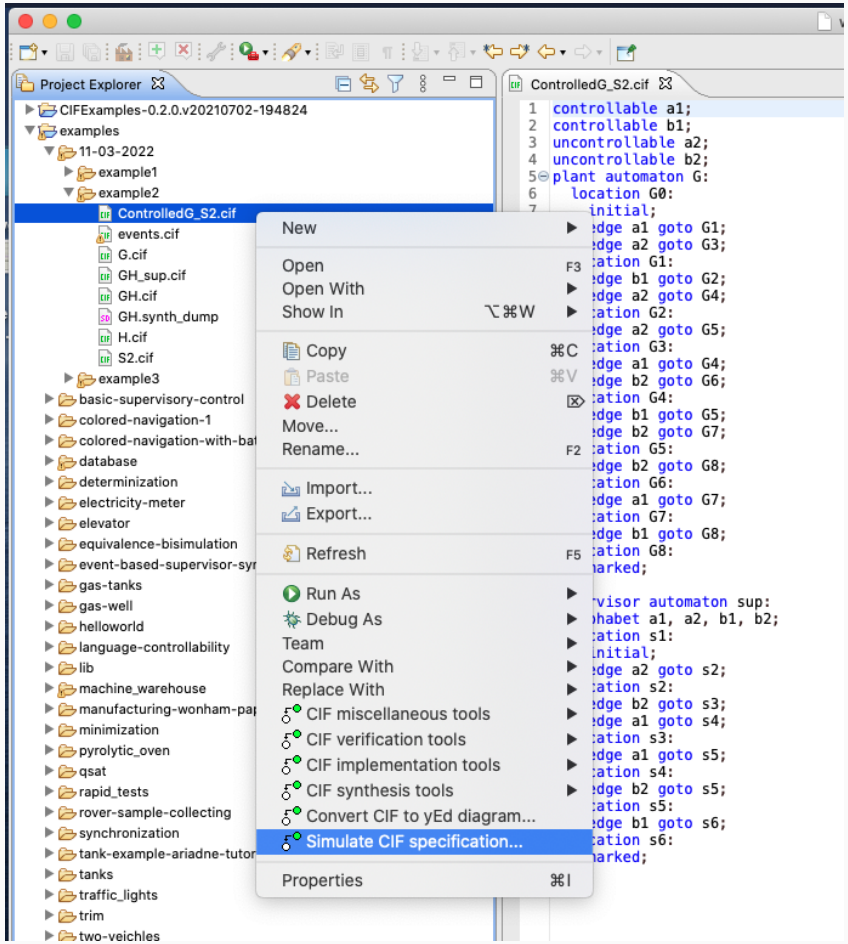
Example 2 - Supervisor Deployment - ControlledG_S2.cif



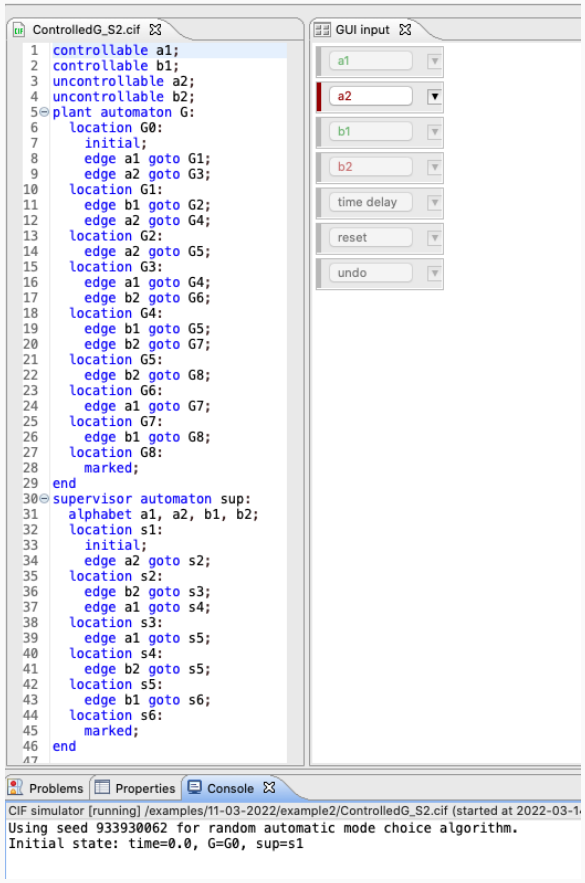

```
uncontrollable a1;
uncontrollable a2;
controllable b1;
controllable b2;

plant automaton G:
  location G0:
    initial;
  ...
  location G8:
    marked;
end

supervisor automaton S2:
  alphabet a1, a2, b1, b2;
  location s1:
    initial;
  ...
  location s10:
    marked;
end
```

Example 2 - Simulation of the Controlled Plant

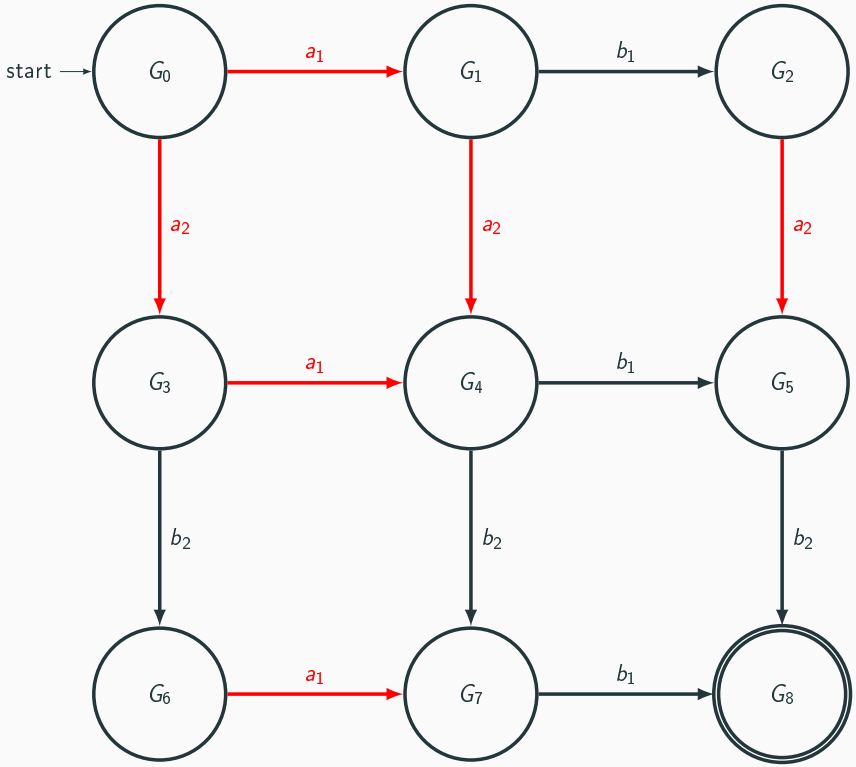
Simulation Menu



Interactive Simulation



Example 3 - Events, Plant, Requirement



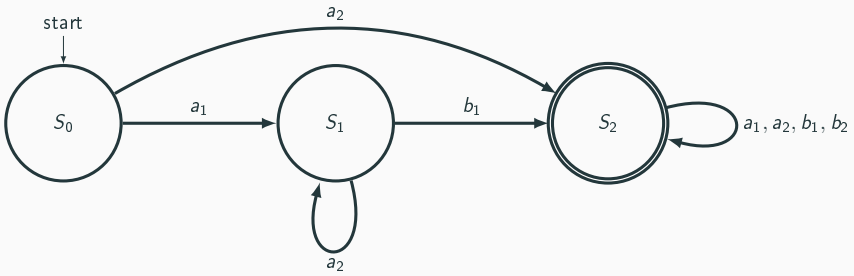
- Events a_1, a_2 are uncontrollable
- Events b_1, b_2 are controllable

```
uncontrollable a1, a2;
controllable b1, b2;
```

- **The `event.cif` file is the same of that given in Example 1**
- **Plant file `G.cif` is still the same**

Example 3 - Decomposition - R1.cif

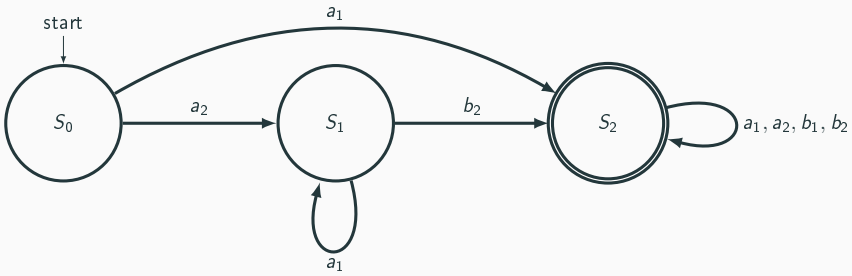
Essential Requirement R_1




```
import "events.cif";

requirement R1:
    location S0: initial;
        edge a1 goto S1;
        edge a2 goto S2;

    location S1:
        edge a2;
        edge b1 goto S2;

    location S2: marked;
        edge a1,a2,b1,b2;
end
```

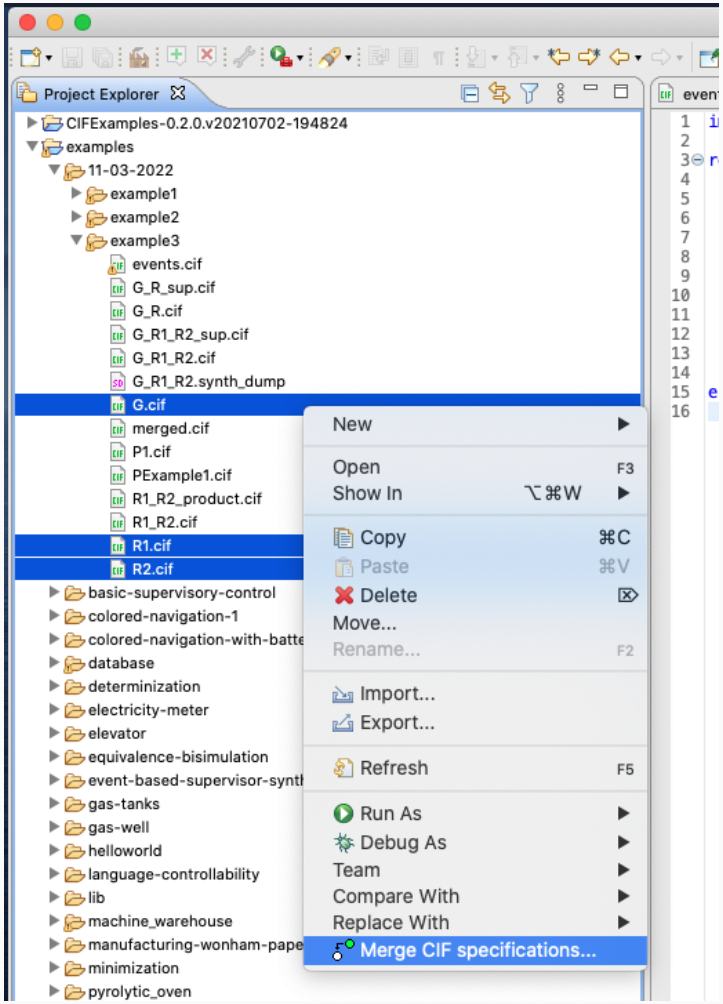
Example 3 - Decomposition - R2.cif

Essential Requirement R_2




```
import "events.cif";

requirement R2:
    location S0: initial;
        edge a2 goto S1;
        edge a1 goto S2;

    location S1:
        edge a1;
        edge b2 goto S2;

    location S2: marked;
        edge a1,a2,b1,b2;
end
```

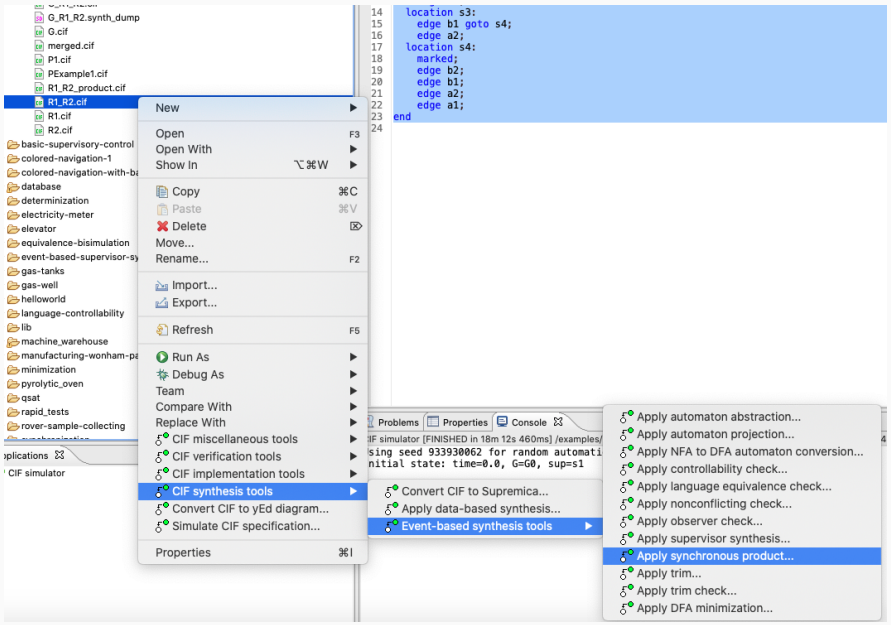
Example 2 - Merge - G_R1_R2.cif



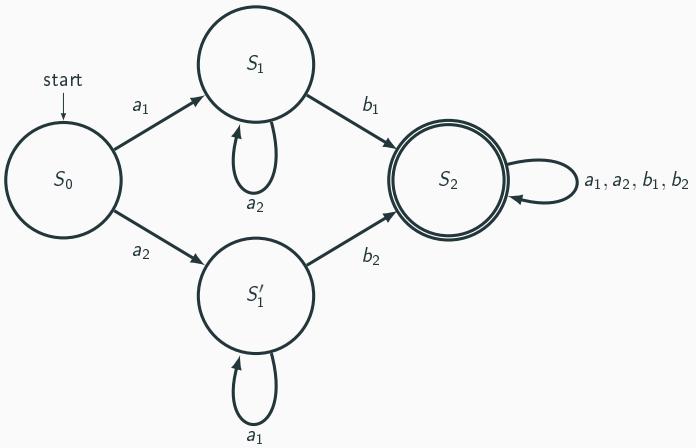

```
uncontrollable a1;
uncontrollable a2;
controllable b1;
controllable b2;

plant automaton G:
  location G0: initial;
    ...
end

requirement automaton R1:
  location S0: initial;
    ...
end

requirement automaton R2:
  location S0: initial;
    ...
end
```

Then, the supervisor is synthesized and deployed the same way

Example 3 - Product of Two Automata - $R := R_1 \| R_2$



Essential Requirement $R := R_1 \| R_2$

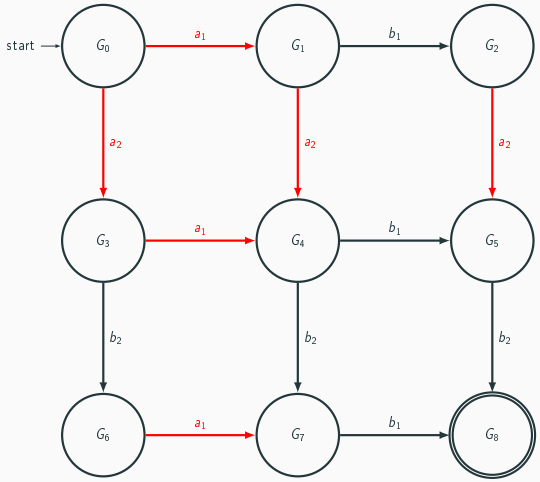


Then, merge, synthesize, and deploy.

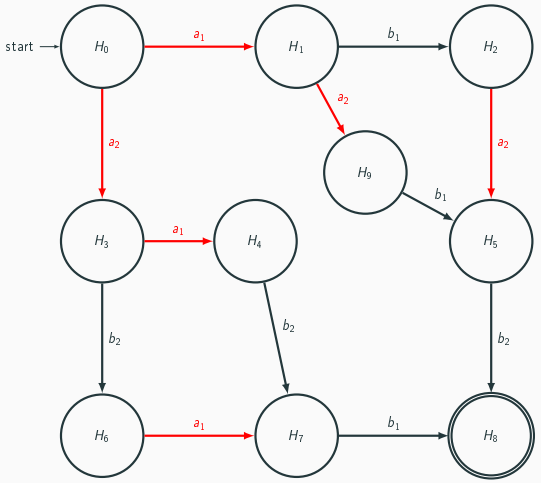
```
uncontrollable a1;
uncontrollable a2;
controllable b1;
controllable b2;

requirement automaton R:
  alphabet a1, a2, b1, b2;
  location s1:
    initial;
    edge a2 goto s2;
    edge a1 goto s3;
  location s2:
    edge b2 goto s4;
    edge a1;
  location s3:
    edge b1 goto s4;
    edge a2;
  location s4:
    marked;
    edge b2, b1, a2, a1;
end
```

Equivalence of Requirements on the Same Plant

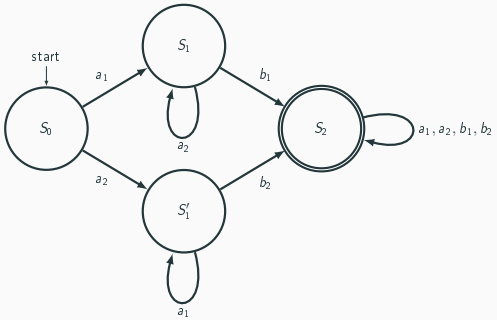
Plant G



Full Requirement H



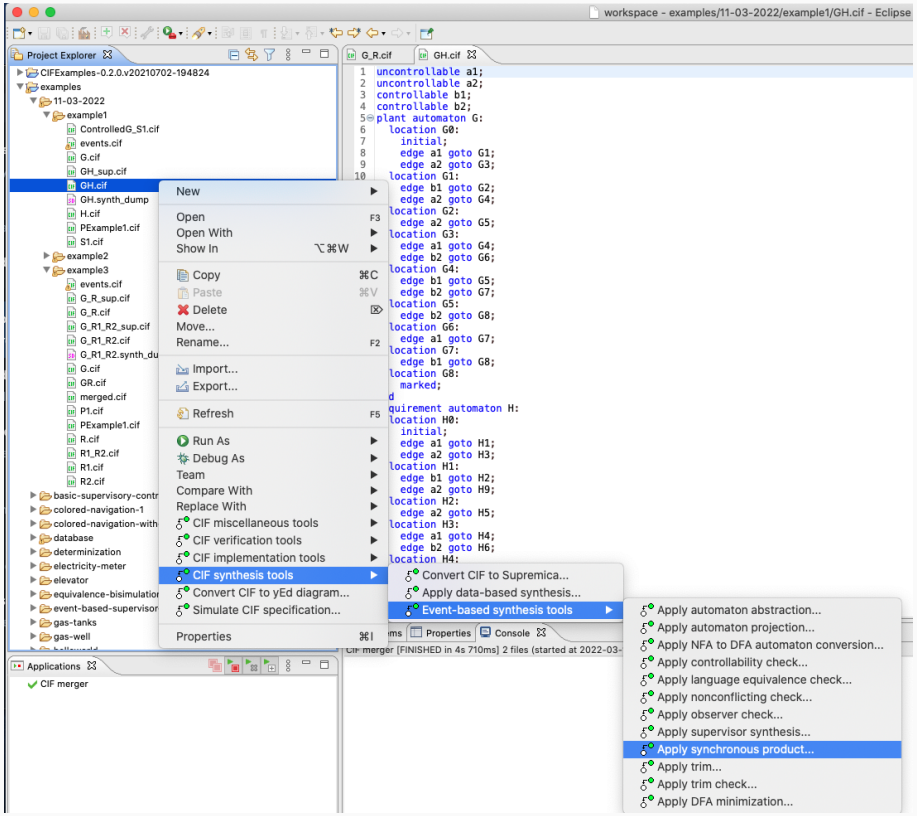
Essential Requirement R



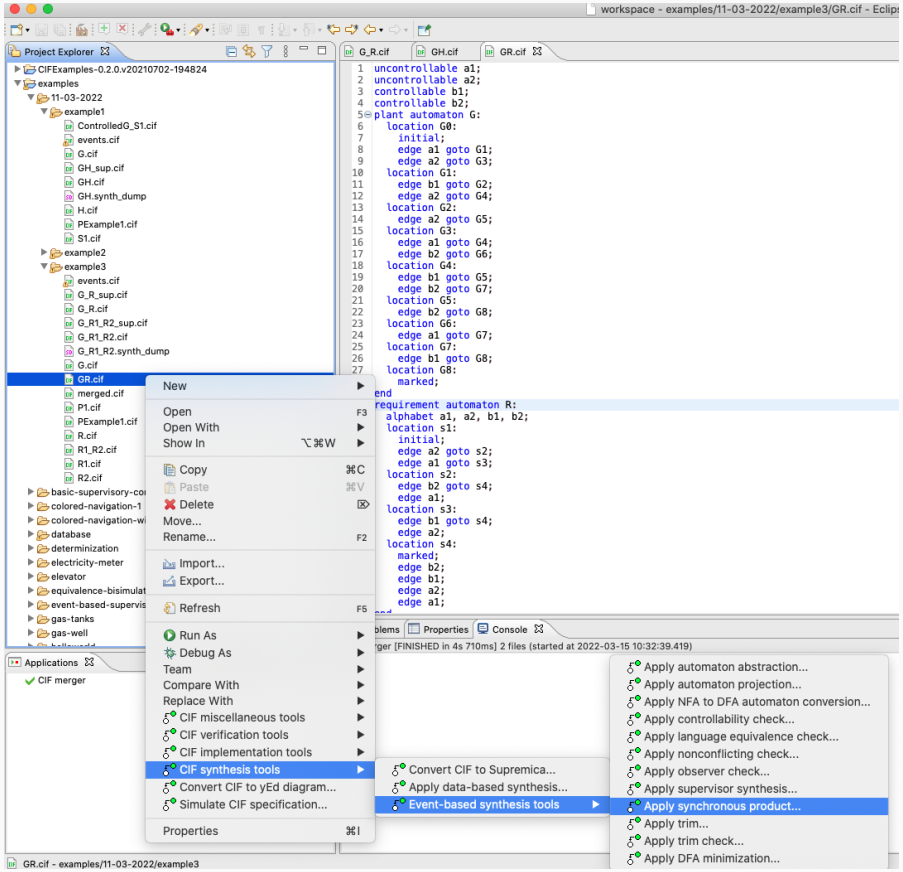
The effect of H on G is the same of the effect of R on G if and only if $G \| H$ is equivalent to $G \| R$.

Equivalence of Requirements on the Same Plant

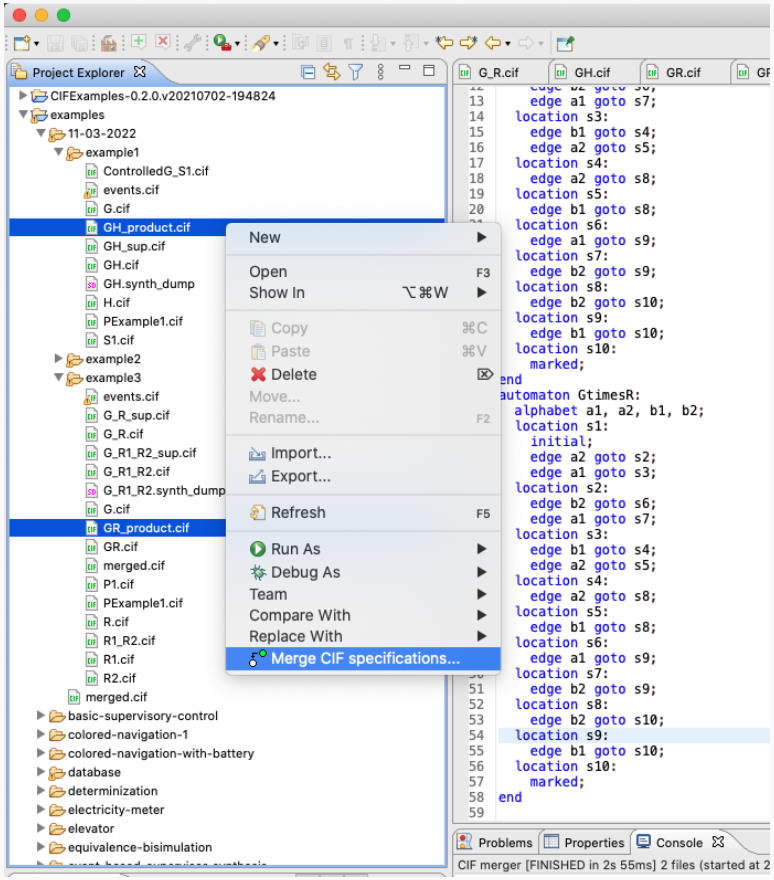
Example 1: $G \| H$



Example 3: $G \| R$

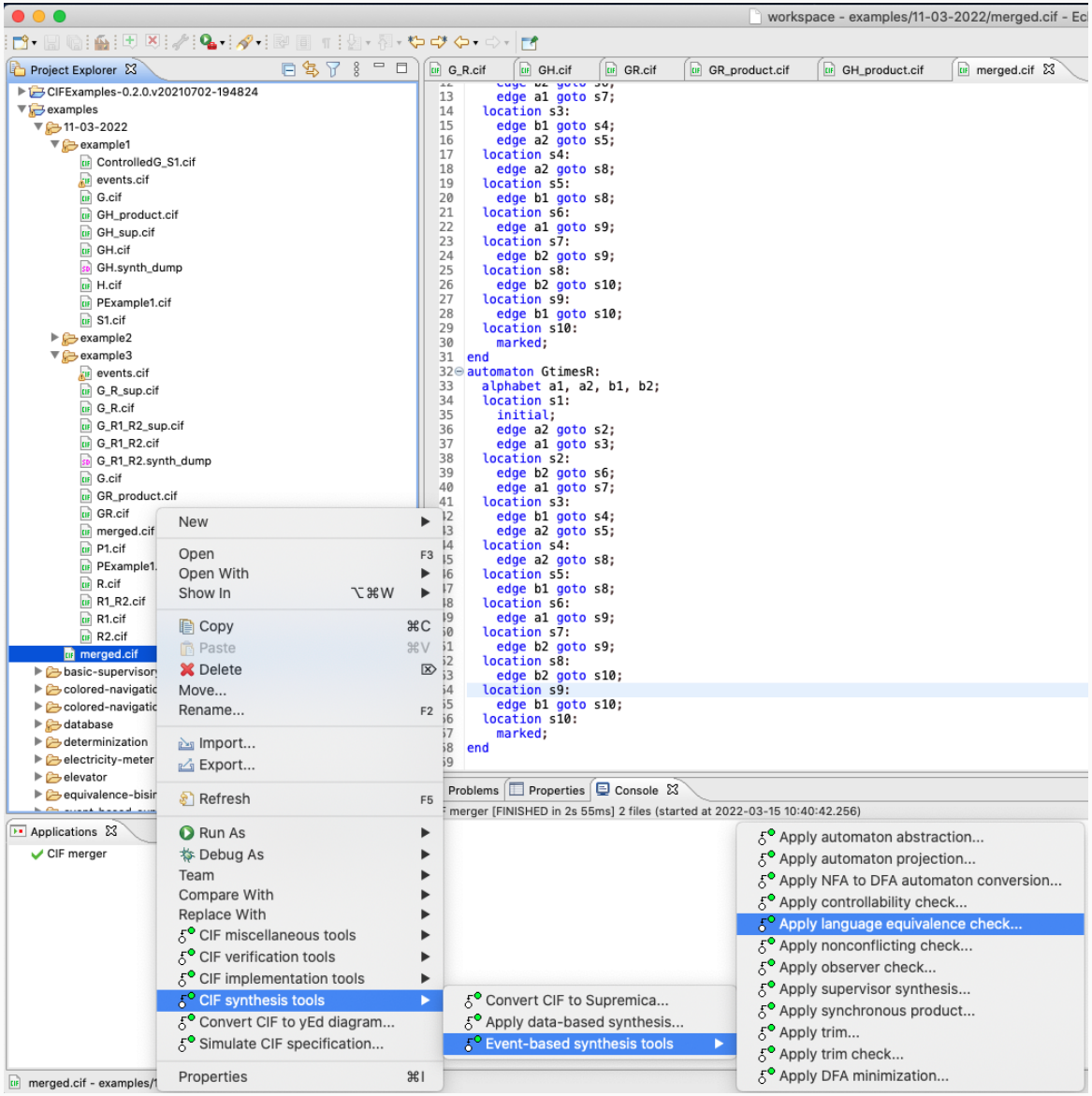


Example 3 - Merge - $G\|H$ and $G\|R$



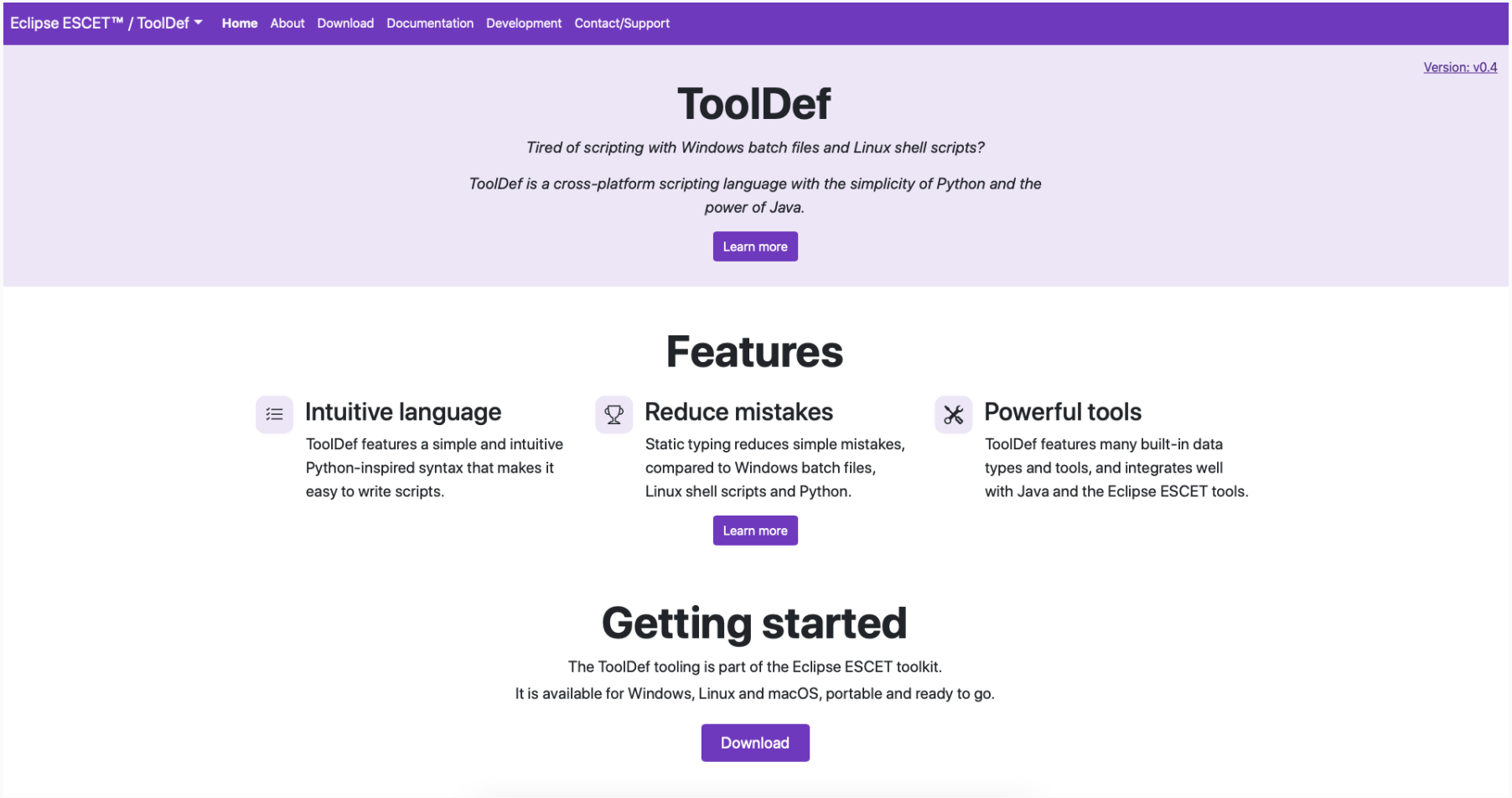

```
uncontrollable a1;
uncontrollable a2;
controllable b1;
controllable b2;
automaton GtimesH:
  alphabet a1, a2, b1, b2;
  location s1:
    initial;
    ...
  location s10:
    marked;
end
automaton GtimesR:
  alphabet a1, a2, b1, b2;
  location s1:
    initial;
    ..
  location s10:
    marked;
end
```

Example 3 - Language Equivalence Check

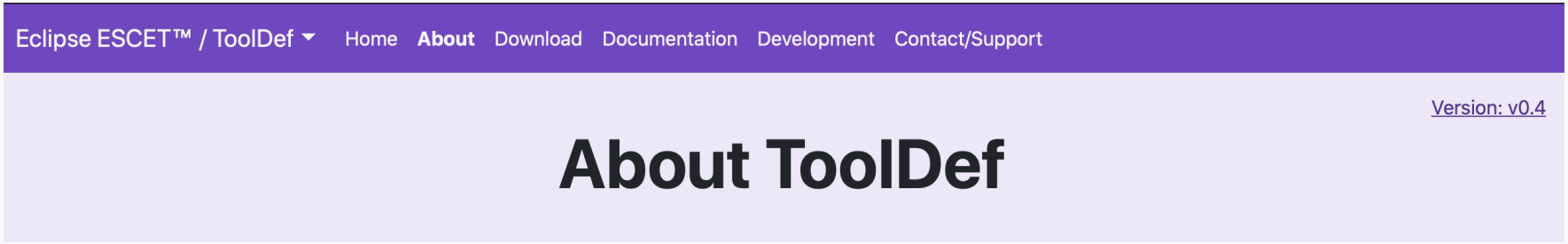


Automata have the same language.

ToolDef



https://www.eclipse.org/escet/tooldef/language-reference/index.html



ToolDef allows us to:

- **write scripts using a simple and intuitive syntax, loosely based on the better aspects of Python.**
- **catch simple mistakes early on due to static typing.**
- **work with data of all kinds, using a large number of built-in data types.**
- **manipulate data and paths, work with files and directories, and much more, with over 80 built-in tools.**
- **share your tools as ToolDef libraries.**
- **unleash the full power of Java by importing any Java static method and using it like any other ToolDef tool.**

ToolDef - First and Most Important

Each script is `File.tooldef`

The first line is:

```
from "lib:cif" import *;
```

ToolDef - Generating Text Files

```
from "lib:cif" import *;
writefile("file.cif", "my text 1\n");
writefile("file.cif", "my text 2\n", append=true);
```

ToolDef - Generating Text Files - Alternative Way

```
from "lib:cif" import *;
string text1 = "my text";
string text2 = "my text";
writefile("file.cif", text1);
writefile("file.cif", text2, append=true);
```

ToolDef - Merging CIF files

Help screen

```
cifmerge("-h");
```

Simple usage

```
cifmerge(
  "InputFile1.cif",
  "InputFile2.cif",
  ...
  "InputFileN.cif"
);
```

Specifying the output file

```
cifmerge(
  "InputFile1.cif",
  "InputFile2.cif",
  ...
  "InputFileN.cif",
  "-o OutputFile.cif"
);
```

The CIF merger can be used to merge two or more CIF specifications into a single CIF specification

https://www.eclipse.org/escet/cif/tools/mergecif.html

ToolDef - Synthesizing a Supervisor

Help screen

```
cifsupsynth("-h");
```

Simple usage

```
cifsupsynth(
  "InputFile.cif",
);
```

Specifying the output file

```
cifsupsynth(
  "InputFile.cif",
  "-o OutputFile"
);
```

Specifying a name for the OutputFile

```
cifsupsynth(
  "InputFile.cif",
  "-n Supervisor.cif",
  "-o OutputFile.cif"
);
```

Enabling the Debug

```
cifsupsynth(
  "InputFile.cif",
  "-n Supervisor.cif",
  "-o OutputFile.cif"
  "-d DebugFile.synth_dump"
);
```

The tool takes a .cif file containing plant and requirement automata.

https://www.eclipse.org/escet/cif/tools/eventbased/supervisorsynthesis.html

Debug (synthesis analysis):

https://www.eclipse.org/escet/cif/tools/eventbased/synthesis-analysis.html

ToolDef - Deploying a Supervisor

Intuitive use

```
cifmerge(
  "Plant.cif",
  "Supervisor.cif",
  "-o ControlledPlant.cif"
);
```

General use

```
cifmerge(
  "Plant1.cif",
  ...
  "PlantN.cif",
  "Supervisor1.cif",
  ...
  "SupervisorN.cif",
  "-o ControlledPlant.cif"
);
```

Simply Merge the Plant with the Supervisor

ToolDef - Parallel Composition

Help screen

```
cifprod("-h");
```

Simple usage

```
cifprod(
  "InputFile.cif",
);
```

Specifying the output file

```
cifprod(
  "InputFile.cif",
  "-o OutputFile.cif"
);
```

Specifying the output file and a name

```
cifprod(
  "InputFile.cif",
  "-n Name",
  "-o OutputFile.cif"
);
```

The tool takes a .cif file containing all automata to combine, and produces a new .cif file with the product automaton

https://www.eclipse.org/escet/cif/tools/eventbased/product.html

ToolDef - Language Equivalence Check

Help screen

```
ciflngeqv("-h");
```

Usage

```
ciflngeqv(
  "InputFile.cif",
);
```

The tool takes a .cif file containing exactly two automata, that must be deterministic, have the same alphabet, and have an initial location

https://www.eclipse.org/escet/cif/tools/eventbased/language-equivalence-check.html

ToolDef - Executing ToolDef Scripts From ToolDef Scripts

Help screen

```
tooldef("-h");
```

Usage

```
tooldef(
  "Script.tooldef",
);
```

The tool takes a `.tooldef` and executes it

ToolDef - Overview of Scriptable Tools

Have a look at

https://www.eclipse.org/escet/cif/tools/scripting/tools.html

for more scriptable tools.