Lecture 3, part 2: Sun Graphs and Execution Trees

Knowledge and Gossip — ESSLLI 2022

Malvin Gattinger (ILLC, Amsterdam)

2022-08-10, Galway

https://malv.in/2022/gossip/

Sun graphs

Sun graphs in ElmGossip

Examples:

- Total graphs are sun graphs!

  `Abc aBc abC`

Sun graphs in ElmGossip

Examples:

- Total graphs are sun graphs!

  `Abc aBc abC`

- What is a minimal example that is *not* a sun graph? 🤔

  `Ab Bc C`

Sun graphs in ElmGossip

Examples:

- Total graphs are sun graphs!

  `Abc aBc abC`

- What is a minimal example that is *not* a sun graph? 

  `Ab Bc C`

- One more secret is enough to make it a sun again:

  `Ab Bc Ca`

Sun graphs in ElmGossip

Examples:

- Total graphs are sun graphs!

  `Abc aBc abC`

- What is a minimal example that is *not* a sun graph? 

  `Ab Bc C`

- One more secret is enough to make it a sun again:

  `Ab Bc Ca`

- Check the characterization “LNS is strongly succ iff sun graph” with these examples!

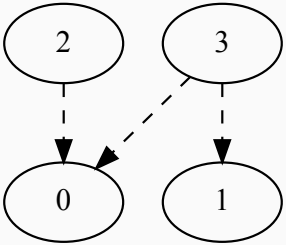
How does ElmGossip check for sun graphs?

```
{-| An initial gossip graph G = (A, N, S) is a sun iff N is strongly connected on the restriction of G
to the set of non-terminal nodes. That is: if you prune terminal nodes (i.e. nodes with only incoming
edges), N should be strongly connected. -}

isSunGraph : Graph Agent Relation -> Bool
isSunGraph graph =
    let
        isTerminal context =
            IntDict.remove context.node.id context.outgoing
                |> IntDict.isEmpty
        prune context prunedGraph =
            if isTerminal context then
                Graph.remove (context.node.id) prunedGraph
            else
                prunedGraph
    in
        Graph.fold prune graph graph
            |> isStronglyConnected Number
```

Execution Trees

Running Example



`nExample`

Execution Trees in GoMoChe

```
GoMoChe> :i tree
tree :: Protocol -> State -> ExecutionTree
```

Execution Trees in GoMoChe

```
GoMoChe> :i tree
tree :: Protocol -> State -> ExecutionTree

GoMoChe> tree lns (nExample,[])
0-1-02-013 I4
(2,0): 02-1-02-013 02-1-02-3
(2,0)(3,0): 0123-1-02-0123 023-1-02-023
(2,0)(3,0)(0,1): 0123-0123-02-0123 0123-0123-02-023
(2,0)(3,0)(0,1)(3,1): 0123-0123-02-0123 0123-0123-02-0123
(2,0)(3,0)(3,1): 0123-0123-02-0123 023-0123-02-0123
(2,0)(3,0)(3,1)(0,1): 0123-0123-02-0123 0123-0123-02-0123
(2,0)(3,1): 02-013-02-013 02-13-02-13
...
```

Show me the tree!

Using the wonderful GraphViz library (https://graphviz.org/) we can also actually draw the tree.

```
pdf $ tree lns (nExample,[])
```

(Not fitting on slide; run it yourself and zoom in to actually see something the resulting picture!)

Execution Trees with epistemic edges

We can further annotate the tree with indistinguishability relation(s).

```
GoMoChe> :i pdfTreeWith
pdfTreeWith :: [Agent] -> Int -> Int -> Protocol -> ExecutionTree -> IO ()
```

(The first `Int` limits the depths of the tree, the second controls the arrangement of nodes.)

Execution Trees with epistemic edges

We can further annotate the tree with indistinguishability relation(s).

```
GoMoChe> :i pdfTreeWith
pdfTreeWith :: [Agent] -> Int -> Int -> Protocol -> ExecutionTree -> IO ()
```

(The first `Int` limits the depths of the tree, the second controls the arrangement of nodes.)

```
GoMoChe> pdfTreeWith [2] 2 1 lns (tree lns (nExample,[]))
```

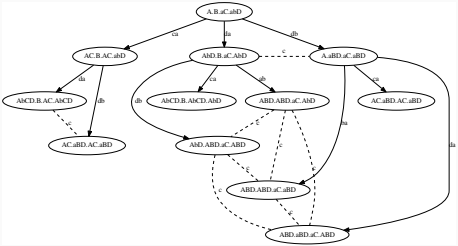
Execution Trees with epistemic edges

We can further annotate the tree with indistinguishability relation(s).

```
GoMoChe> :i pdfTreeWith
pdfTreeWith :: [Agent] -> Int -> Int -> Protocol -> ExecutionTree -> IO ()
```

(The first `Int` limits the depths of the tree, the second controls the arrangement of nodes.)

```
GoMoChe> pdfTreeWith [2] 2 1 lns (tree lns (nExample,[]))
```



Cheap optimization tricks for search problems

- W.l.o.g. fix the first call to be ab.

Cheap optimization tricks for search problems

- W.l.o.g. fix the first call to be ab.
- Fix the first *two* calls to be either ab; bc or to be ab; cd.

  Wait, why is this w.l.o.g.? 

References

- See yesterday!
- See course website for exercises!
- Please report bugs and provide feedback — click here for a form.


Críoch
END