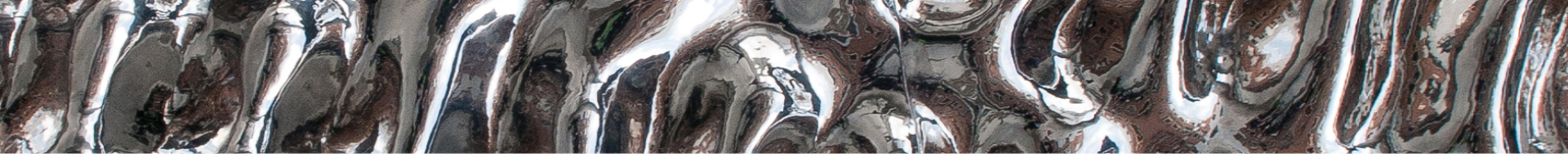


UiO : Department of Informatics
University of Oslo

Abstraction Guided Synthesis of Synchronization

Stein Elgethun

Algorithm overview

- Find a trace that breaks the specification (verification)
- Refine abstraction or modify program?
  - Add an atomicity constraint
  - Refine abstraction
- If no problem traces, implement program!

Simple example program

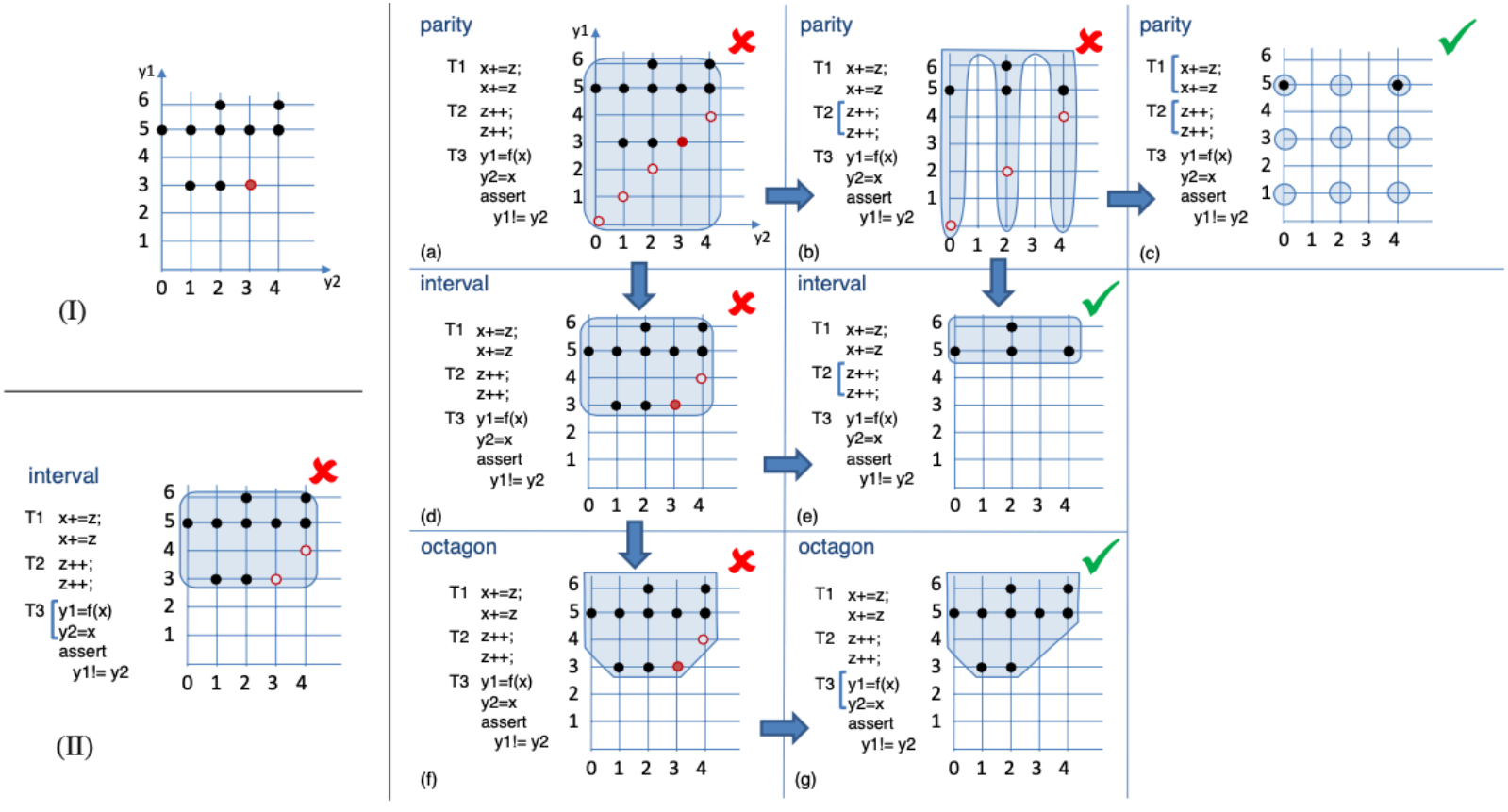
```
T1 {
1: x += z
2: x += z
}
```

```
T2 {
1: z++
2: z++
}
```

```
T3 {
1: y1 = f(x)
2: y2 = x
3: assert
     (y1 ≠ y2)
}
```

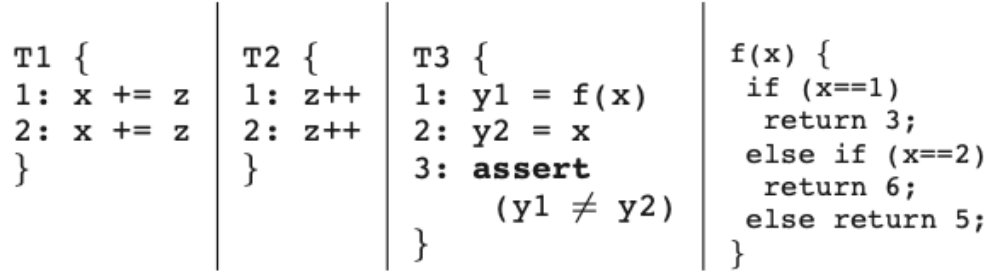
```
f(x) {
 if (x==1)
  return 3;
 else if (x==2)
  return 6;
 else return 5;
}
```

Possible steps in the algorithm



Example program with full information

- Only one invalid trace
  - `z++; x+=z; y1=f(x); z++; x+=z; y2=x; assert`
- Atomicity constraint:
  - `[y1=f(x),y2=x] ∨ [x+=z,x+=z] ∨ [z++,z++]`



Algorithm 1: Abstraction-Guided Synthesis.

Input: Program P, Specification S, Abstraction α
Output: Program satisfying S under α

```
1  φ = true
2  while true do
3      Π = {π | π ∈ [[P]]_α ∩ [[φ]], π ⊭ S}
4      if Π is empty then return implement(P,φ)
5      π = select trace from Π
6      if shouldAvoid(π, α) then
7          ψ = avoid(π)
8          if ψ ≠ false then φ = φ ∧ ψ
9          else abort
10     else
11         α' = refine(α, π)
12         if α' ≠ α then α = α'
13         else abort
14     end
15 end
```

Reference

- Martin Vechev, Eran Yahav, and Greta Yorsh. Abstraction-guided synthesis of synchronization. In POPL, volume 10, pages 327–338, 2010