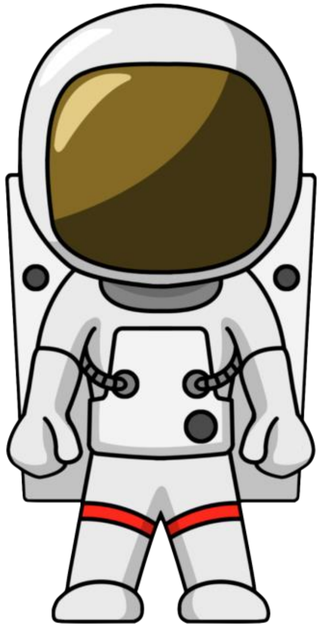
Artifact Evaluated
CAV AEC
Consistent * Complete * Well Documented * Easy to Reuse

Global Guidance for Local Generalization in Model Checking

Hari Govind V K, Yu-Ting Chen,
Sharon Shoham, Arie Gurfinkel

@CAV 2020



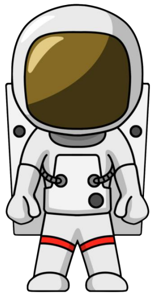
TEL AVIV UNIVERSITY אוניברסיטת תל אביב

UNIVERSITY OF WATERLOO

CHALMERS UNIVERSITY OF TECHNOLOGY

Space Odyssey of Spacer Tom

Engines ON!



- Safety of infinite state systems
  - e.g., sequential programs
  - Generate inductive loop invariants

- IC3-style Model Checking algorithms
  - Generate predecessors to ***Bad*** states (POB)
  - Block them and *generalize* (*lemma*)
  - Stop when you get an invariant

$0 < a < 4 \land b = 4$

$a = b$

$a + b < 4$

```
a = 0;
b = 0;
while (nd()) {
 a++;
 b++;
}
assert (a < 5 ⇒ b < 5);
```

All variables are unbounded integers

`nd()` returns a non deterministic Boolean value.

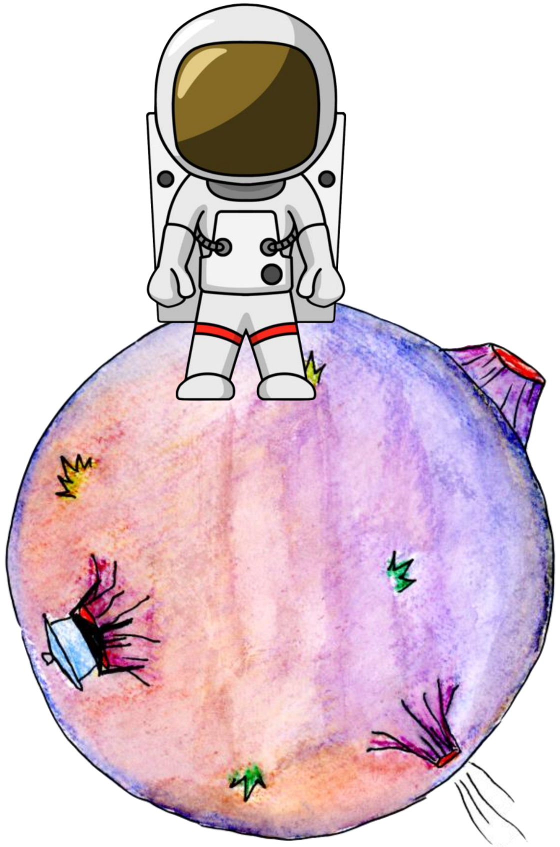
Spacer Tom ONLY knows how to do

Local **reasoning**

- Generalizing from single predecessors
  results in limited exploration horizon

- Generalization typically relies on **interpolation**

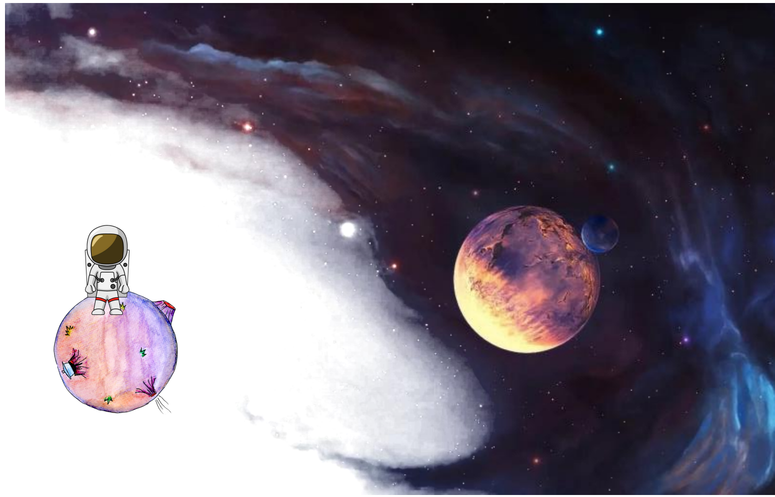
- Interpolation can work wonders!
  e.g., generate breakthrough terms like invariant `a = b`



Ground Control to Spacer Tom:

We've got a PROBLEM!

- Not aware of the structure of the inductive proof so far

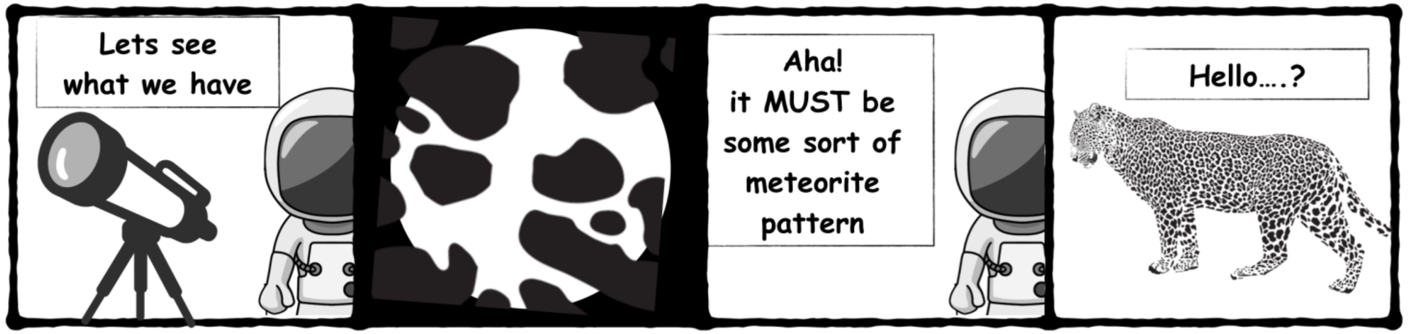


- Interpolant is very much dependent on heuristics in the underlying SMT engine
  - `a + b < 4` is just as likely as `a = b`
- Much more crucial in infinite-state systems than in finite-state systems
  - There are usually infinite generalizations to choose from

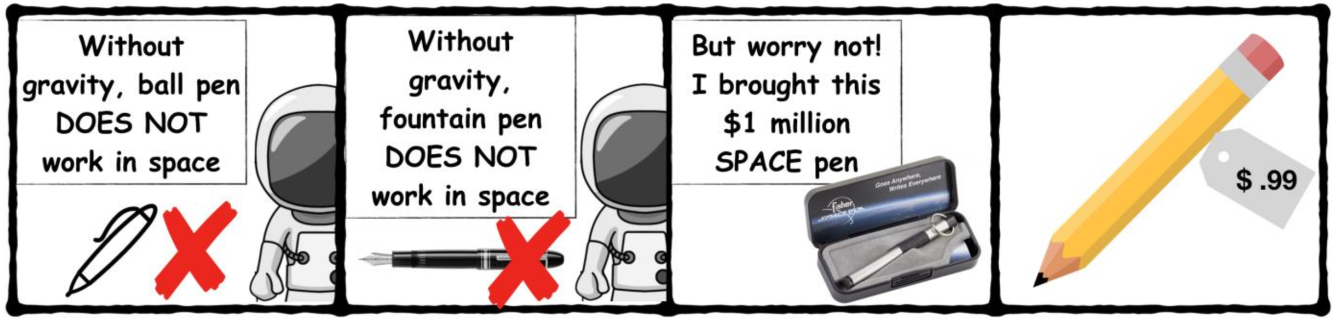
Spacer Tom can be MISSGUIDED!

As illustrated by

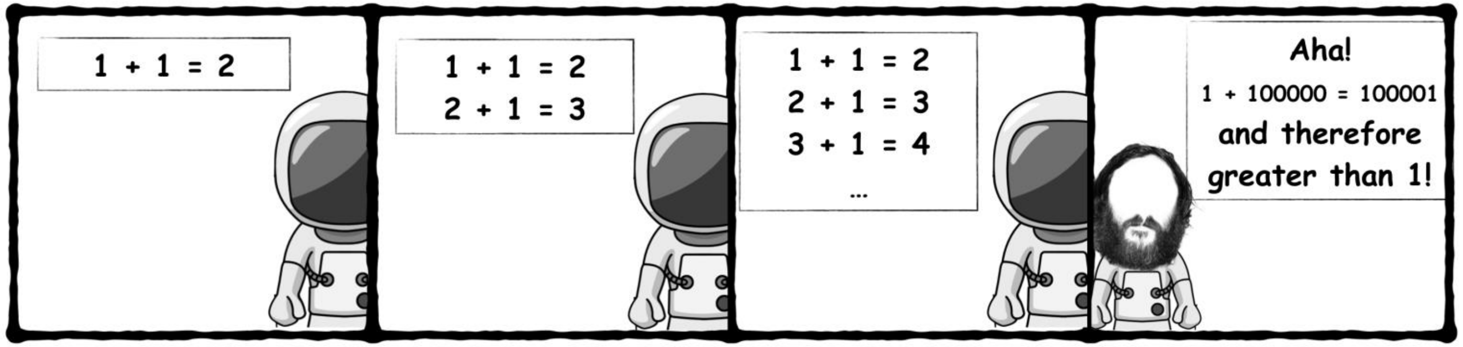
Myopic generalization



Excessive generalization


Without gravity, ball pen DOES NOT work in space
Without gravity, fountain pen DOES NOT work in space
But worry not! I brought this $1 million SPACE pen
$ .99

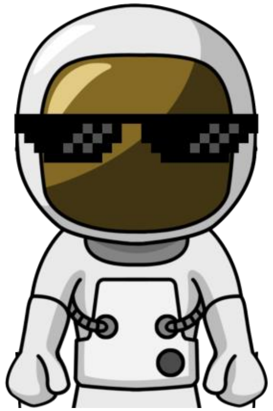
Getting stuck in a rut



Spacer Tom can be MISSGUIDED!

Myopic Generalization

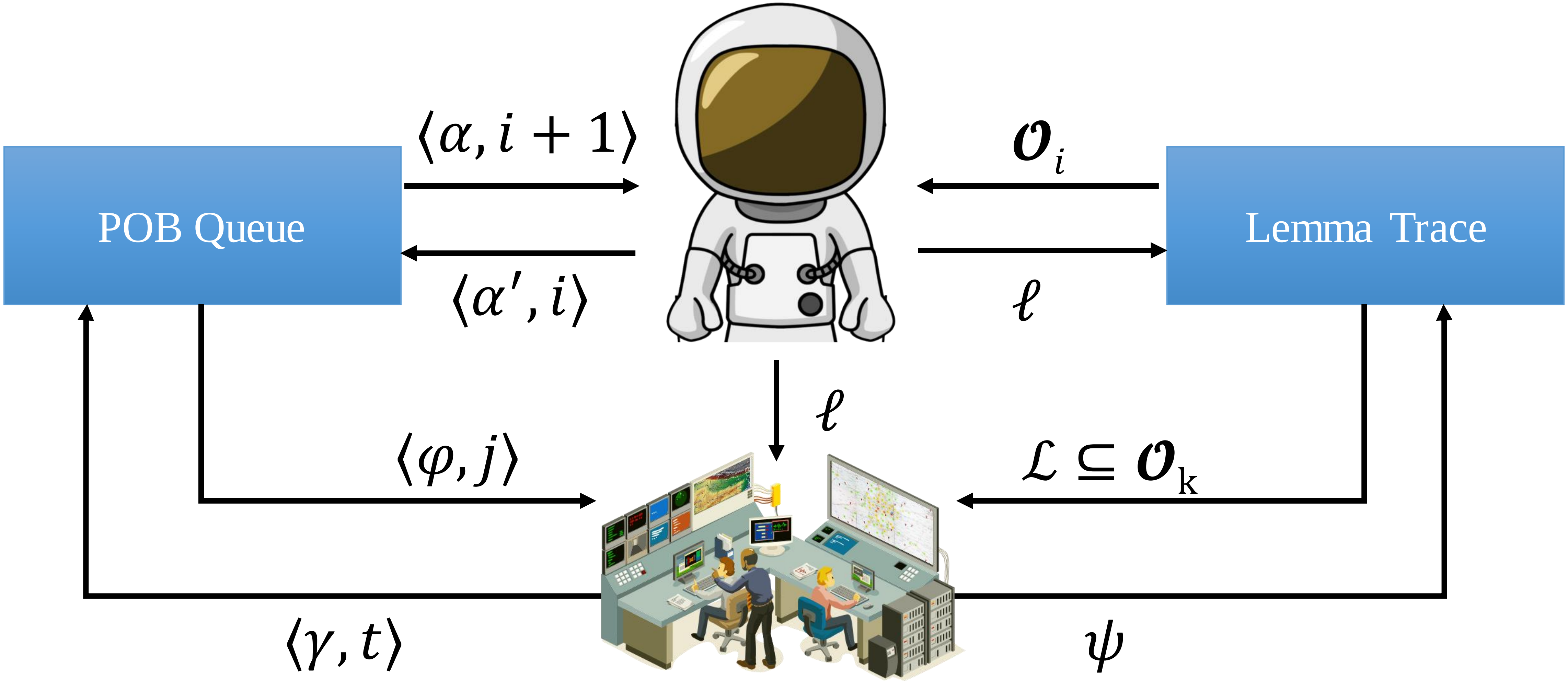
```
a, c = 0;
b, d = 0;
while (nd()) {
inv: (a - c = b – d)
 if (nd()) {a++; b++;}
 else      {c++; d++;}
}
assert (a < c ⇒ b < d);
```



`nd()` returns a non-deterministic Boolean value.

Ground Control to Spacer Tom:

Global Guidance



$\langle \alpha, i+1 \rangle$

$\mathcal{O}_i$

POB Queue

Lemma Trace

$\langle \alpha', i \rangle$

ℓ

ℓ

$\langle \varphi, j \rangle$

$\mathcal{L} \subseteq \mathcal{O}_\mathrm{k}$

$\langle \gamma, t \rangle$

ψ

Ground Control to Spacer Tom:

Global Guidance trinity

Subsume

Concretize

Conjecture

1st Global Guidance to GSpacer Tom:

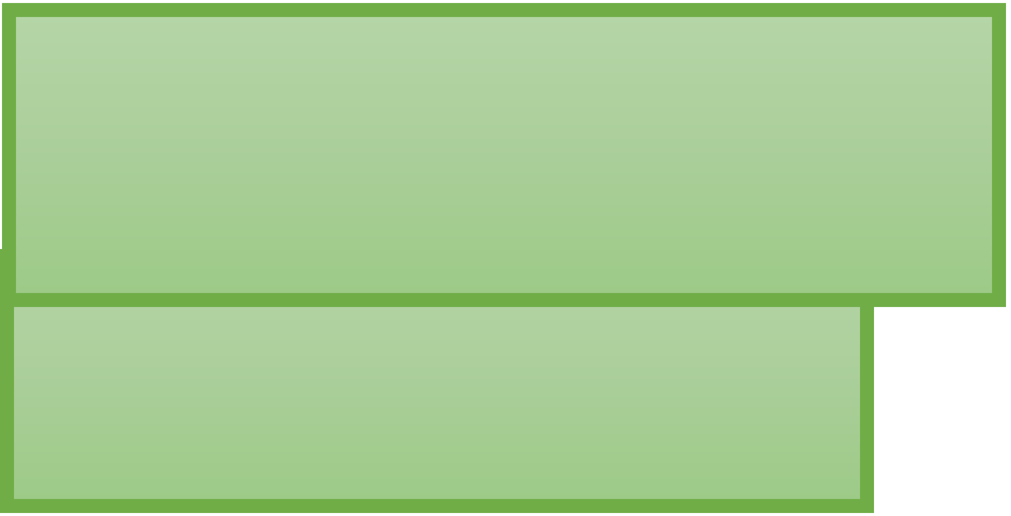
Subsume Rule

```
if ($\exists\psi \cdot \forall\ell \in \mathcal{L} \cdot \psi \Rightarrow \ell$) then
    add $\psi$ to trace
```

1st Global Guidance to GSpacer Tom:

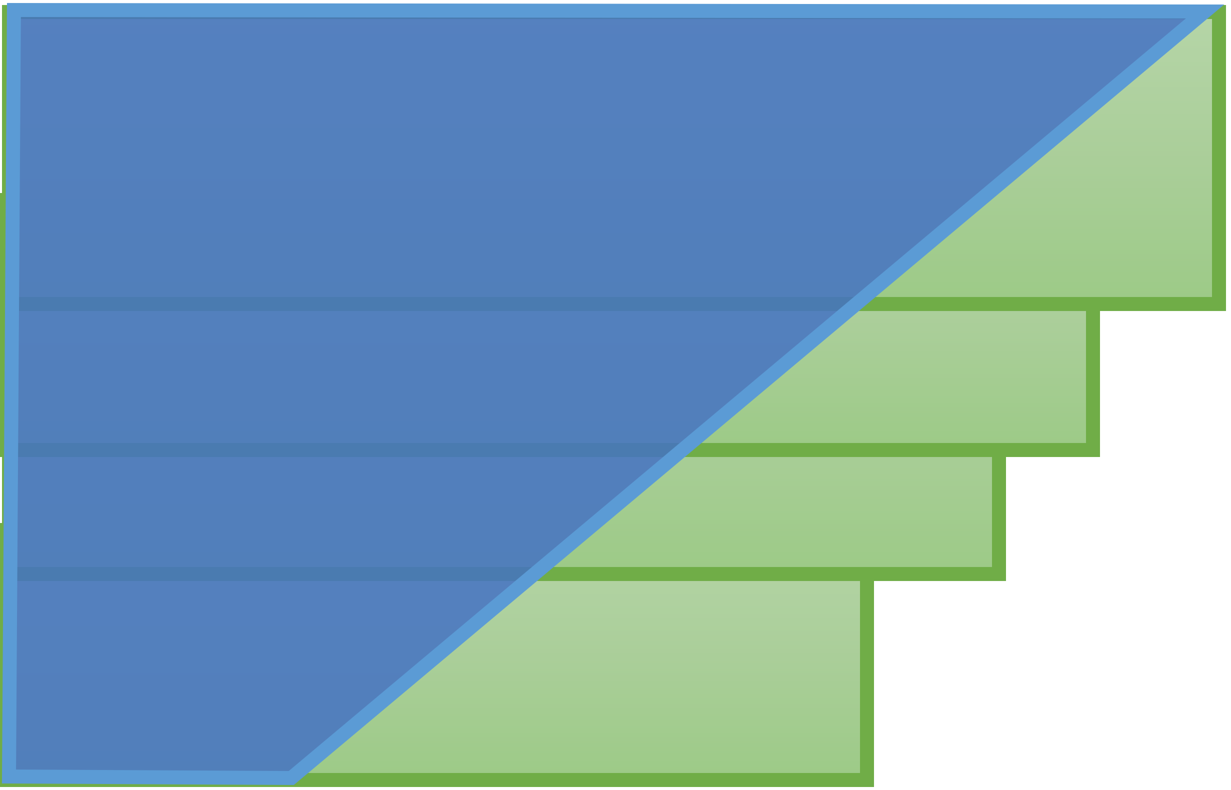
Subsume Rule

```
if $(\exists \psi \cdot \forall \ell \in \mathcal{L} \cdot \psi \Rightarrow \ell)$ then
    add $\psi$ to trace
```



1st Global Guidance to GSpacer Tom:

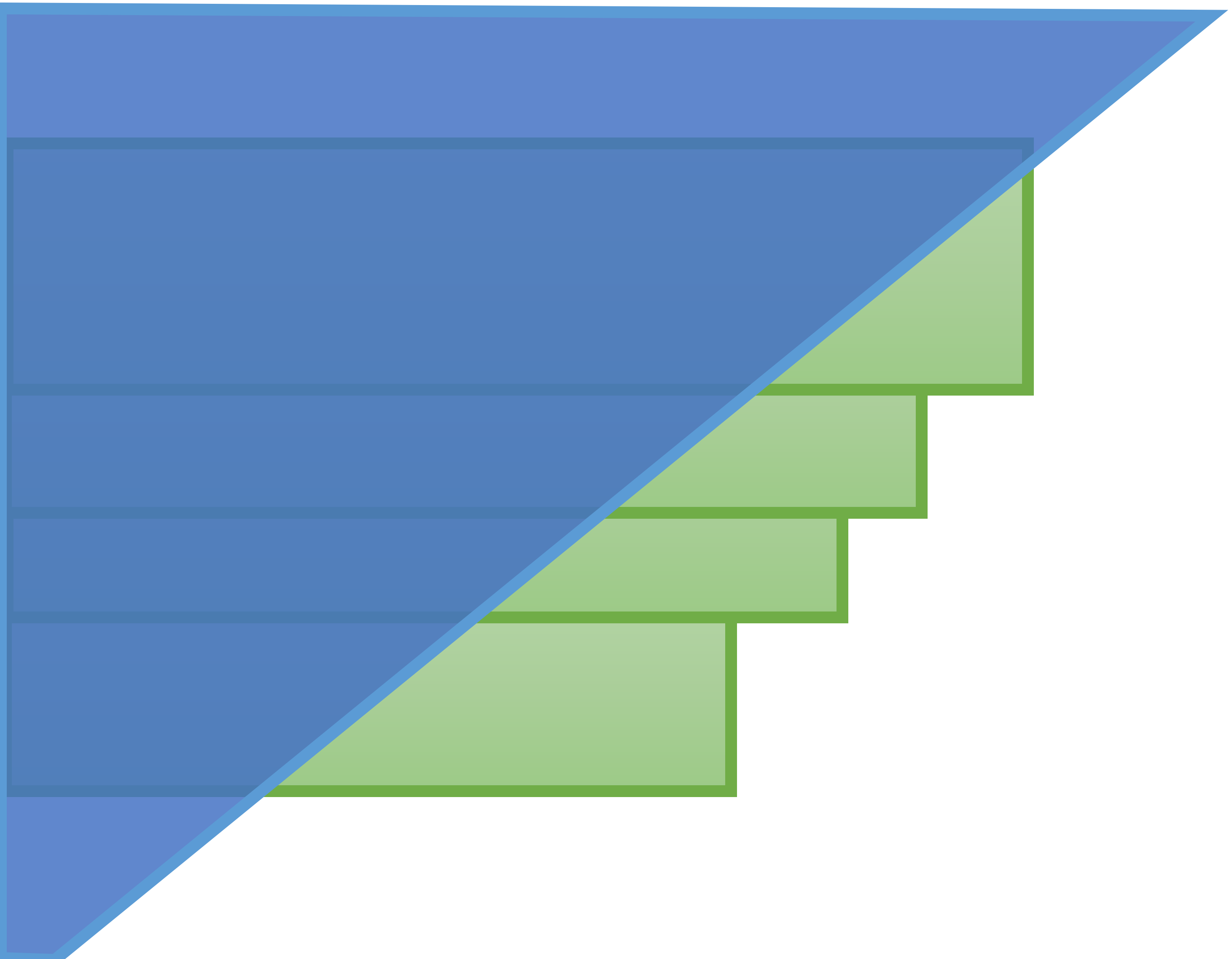
Subsume Rule



```
if ($\exists\psi \cdot \forall\ell \in \mathcal{L} \cdot \psi \Rightarrow \ell$) then
    add $\psi$ to trace
```

1st Global Guidance to GSpacer Tom:

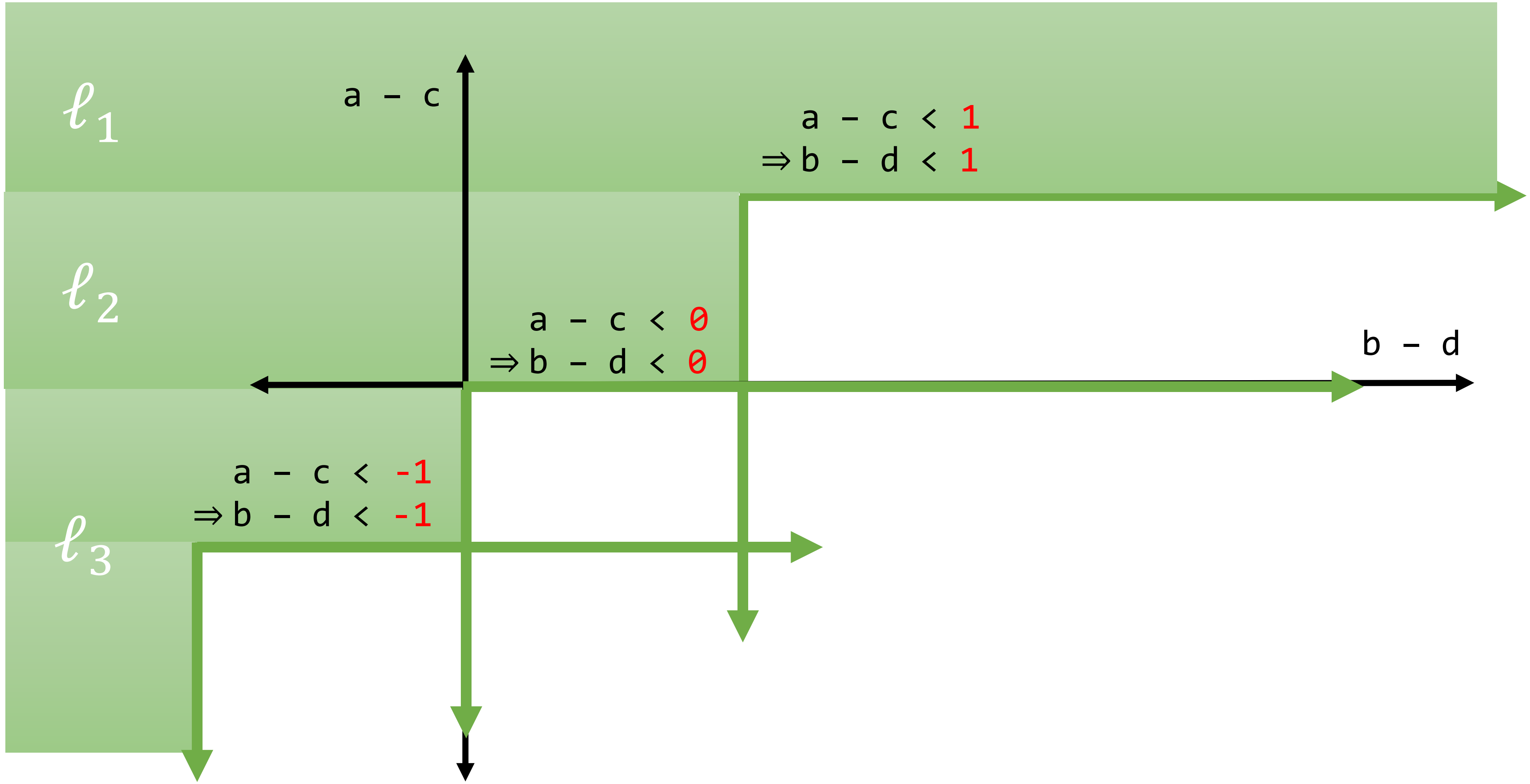
Subsume Rule



```
if $(\exists\psi \cdot \forall\ell \in \mathcal{L} \cdot \psi \Rightarrow \ell)$ then
    add $\psi$ to trace
```

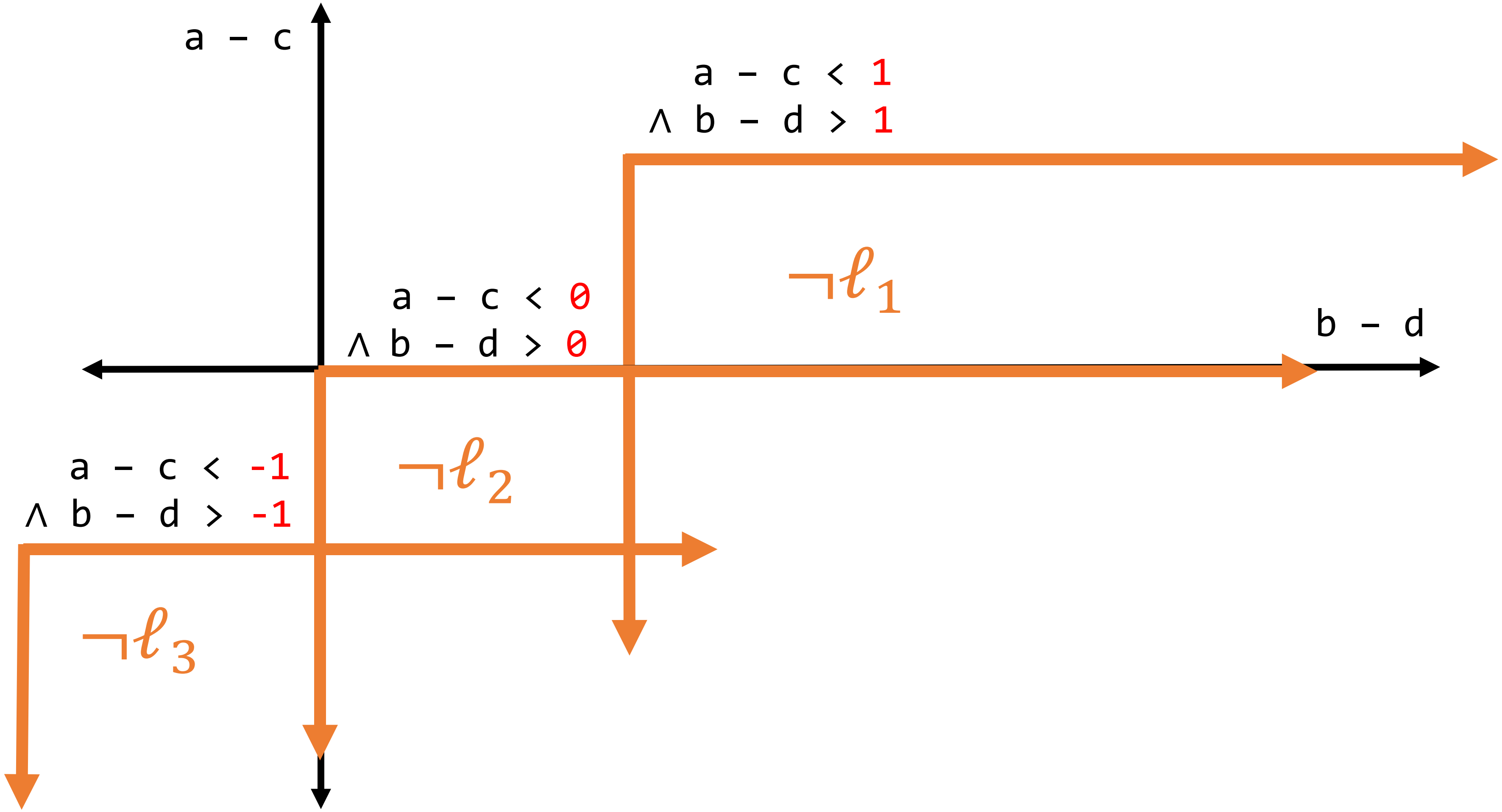
Subsume Rule in Action:

Subsume Rule on LIA



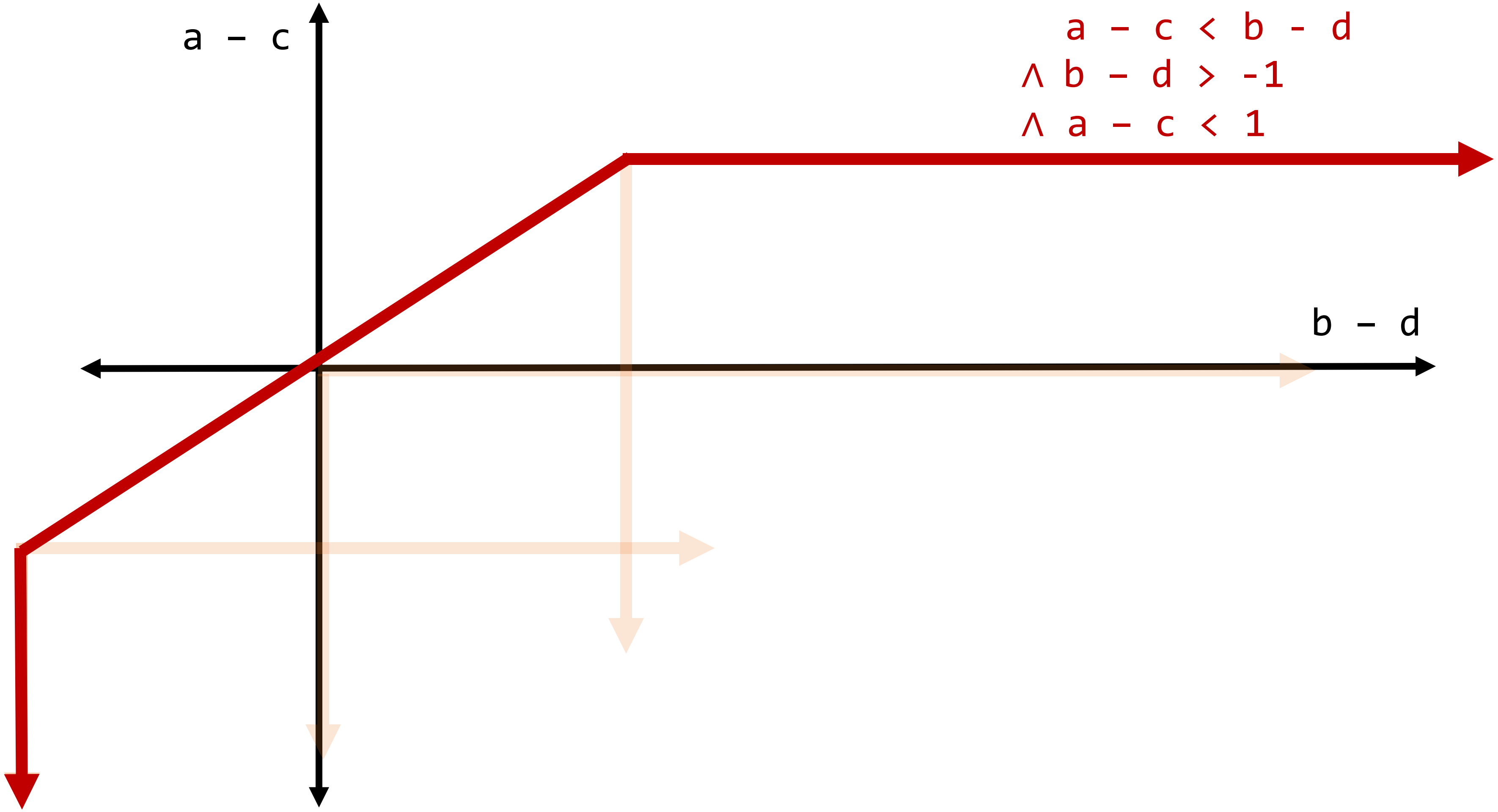
Subsume Rule in Action:

Subsume Rule on LIA



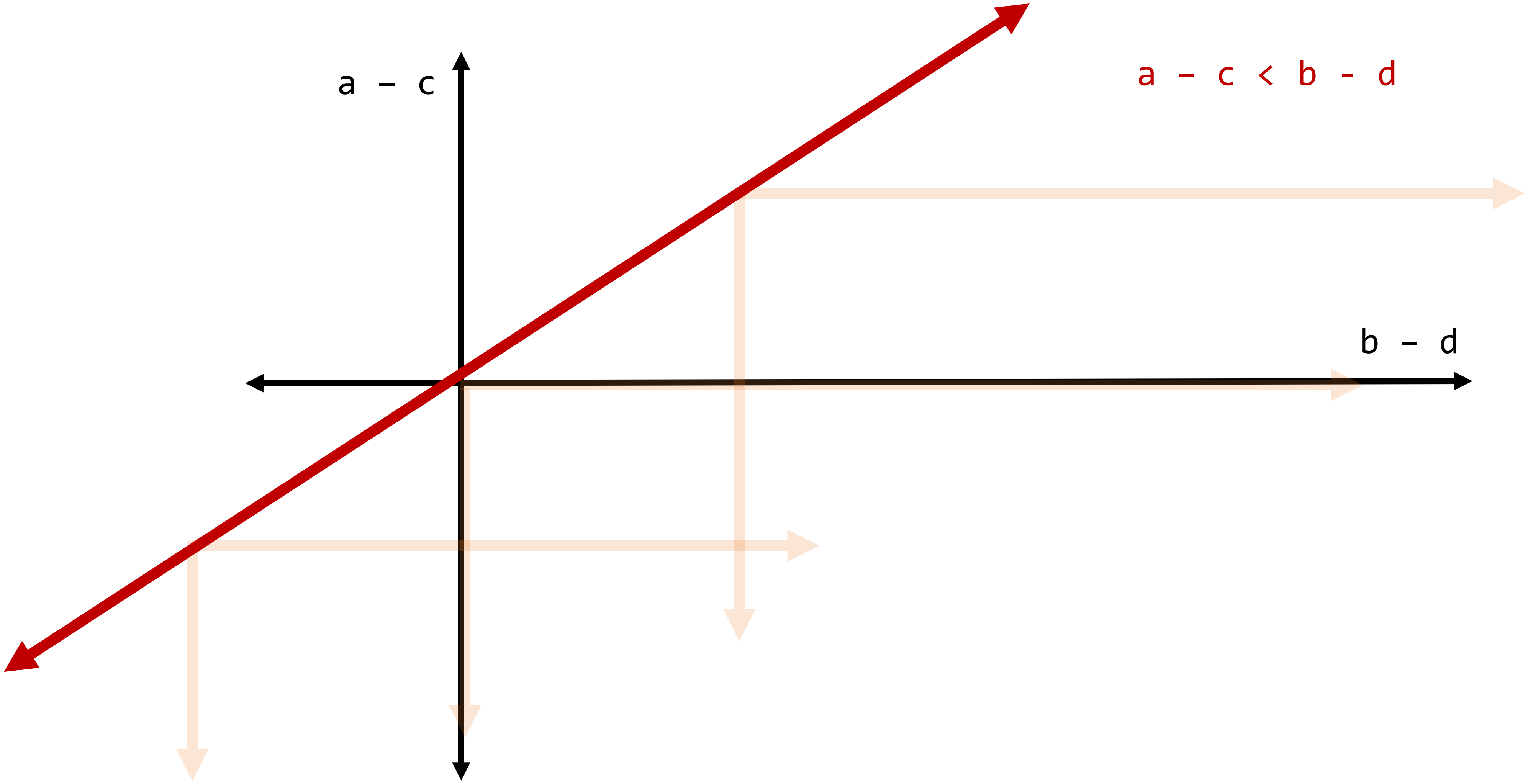
Subsume Rule in Action:

Subsume Rule on LIA



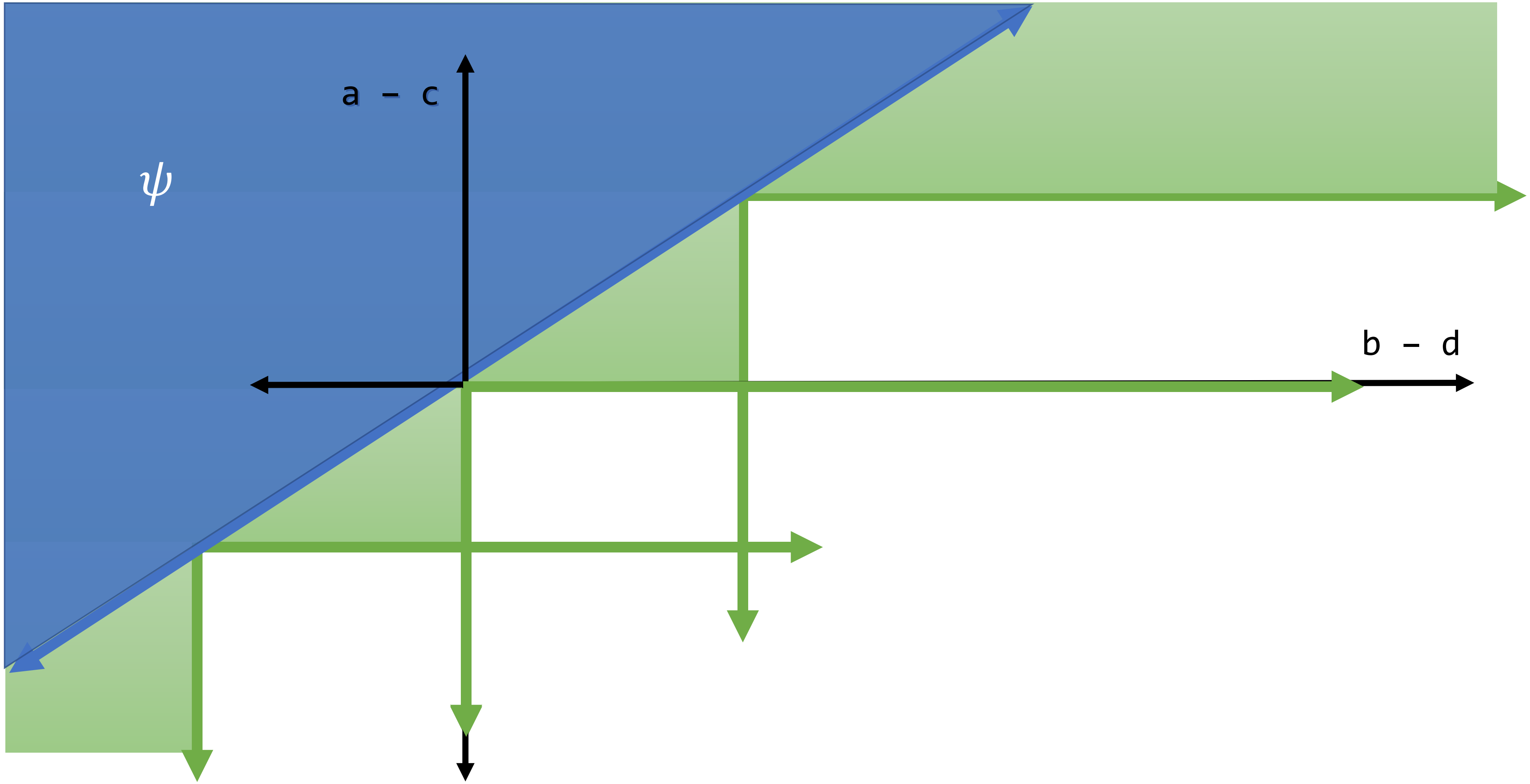
Subsume Rule in Action:

Subsume Rule on LIA



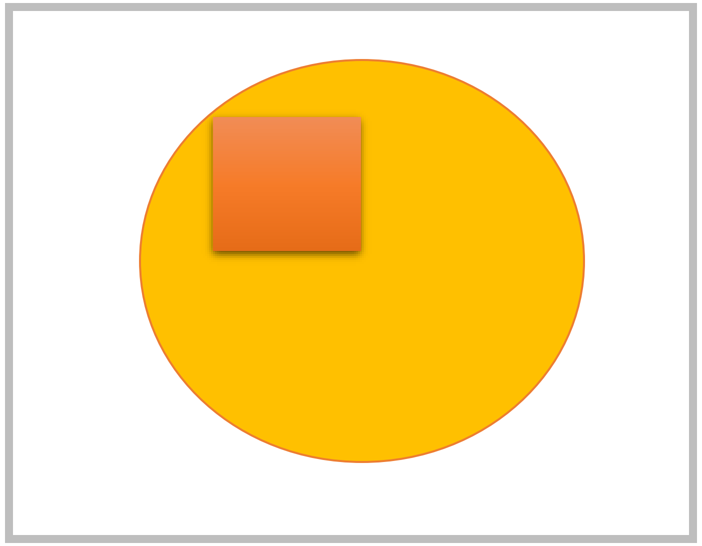
Subsume Rule in Action:

Subsume Rule on LIA



2nd Global Guidance to GSpacer Tom:

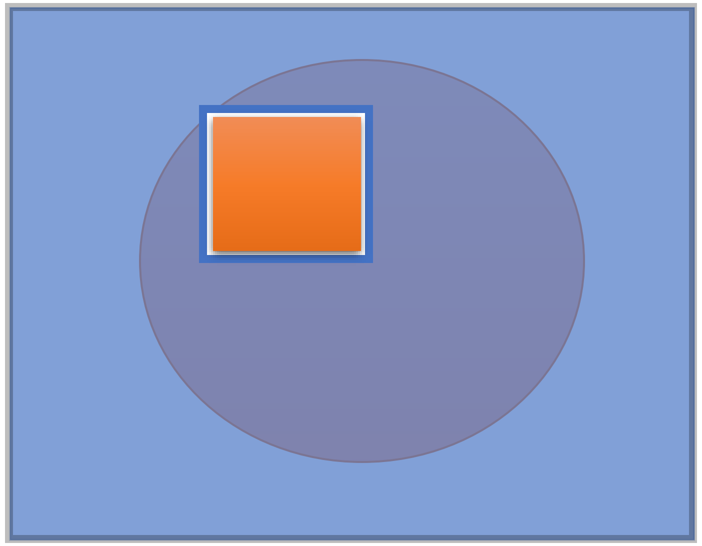
Concretize Rule



```
if $(\forall \ell \in \mathcal{L} \cdot \ell$ partially blocks $\varphi) \wedge$
   $(\exists \gamma \cdot \gamma \Rightarrow \varphi \wedge (\gamma$ is not blocked by $\bigwedge \mathcal{L}))$ then
     add $\gamma$ to POB queue
```

2nd Global Guidance to GSpacer Tom:

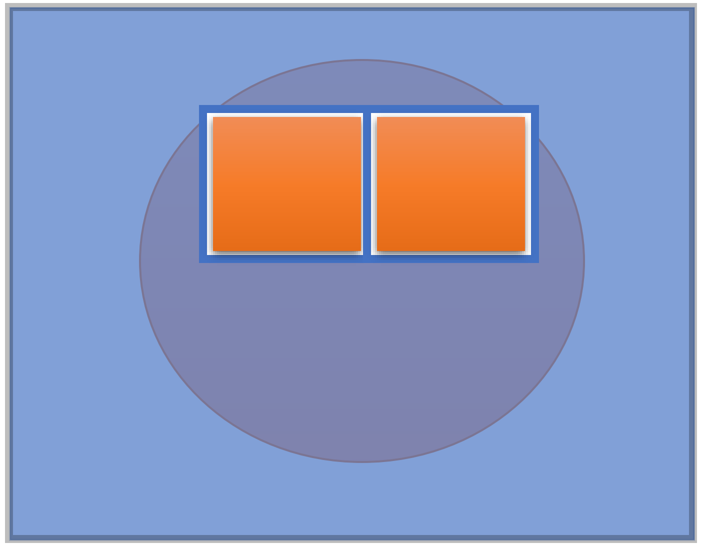
Concretize Rule



```
if $(\forall \ell \in \mathcal{L} \cdot \ell$ partially blocks $\varphi) \wedge$
   $(\exists \gamma \cdot \gamma \Rightarrow \varphi \wedge (\gamma$ is not blocked by $\bigwedge \mathcal{L}))$ then
     add $\gamma$ to POB queue
```

2nd Global Guidance to GSpacer Tom:

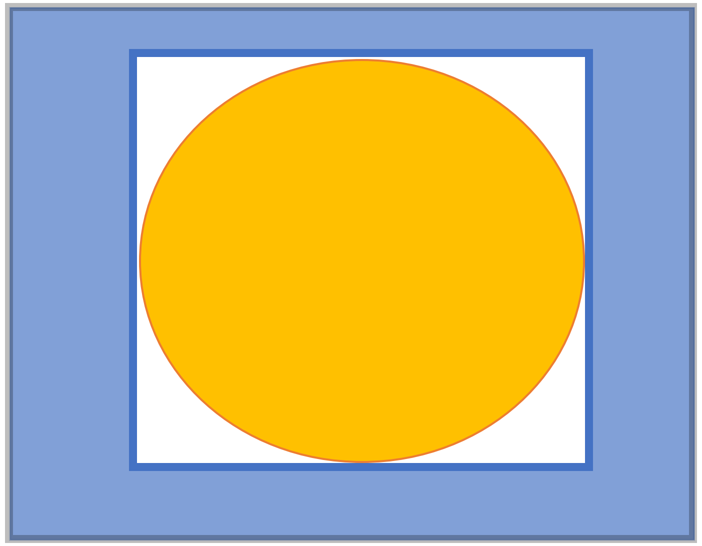
Concretize Rule



```
if $(\forall \ell \in \mathcal{L} \cdot \ell$ partially blocks $\varphi) \wedge$
   $(\exists \gamma \cdot \gamma \Rightarrow \varphi \wedge (\gamma$ is not blocked by $\bigwedge \mathcal{L}))$ then
     add $\gamma$ to POB queue
```

2nd Global Guidance to GSpacer Tom:

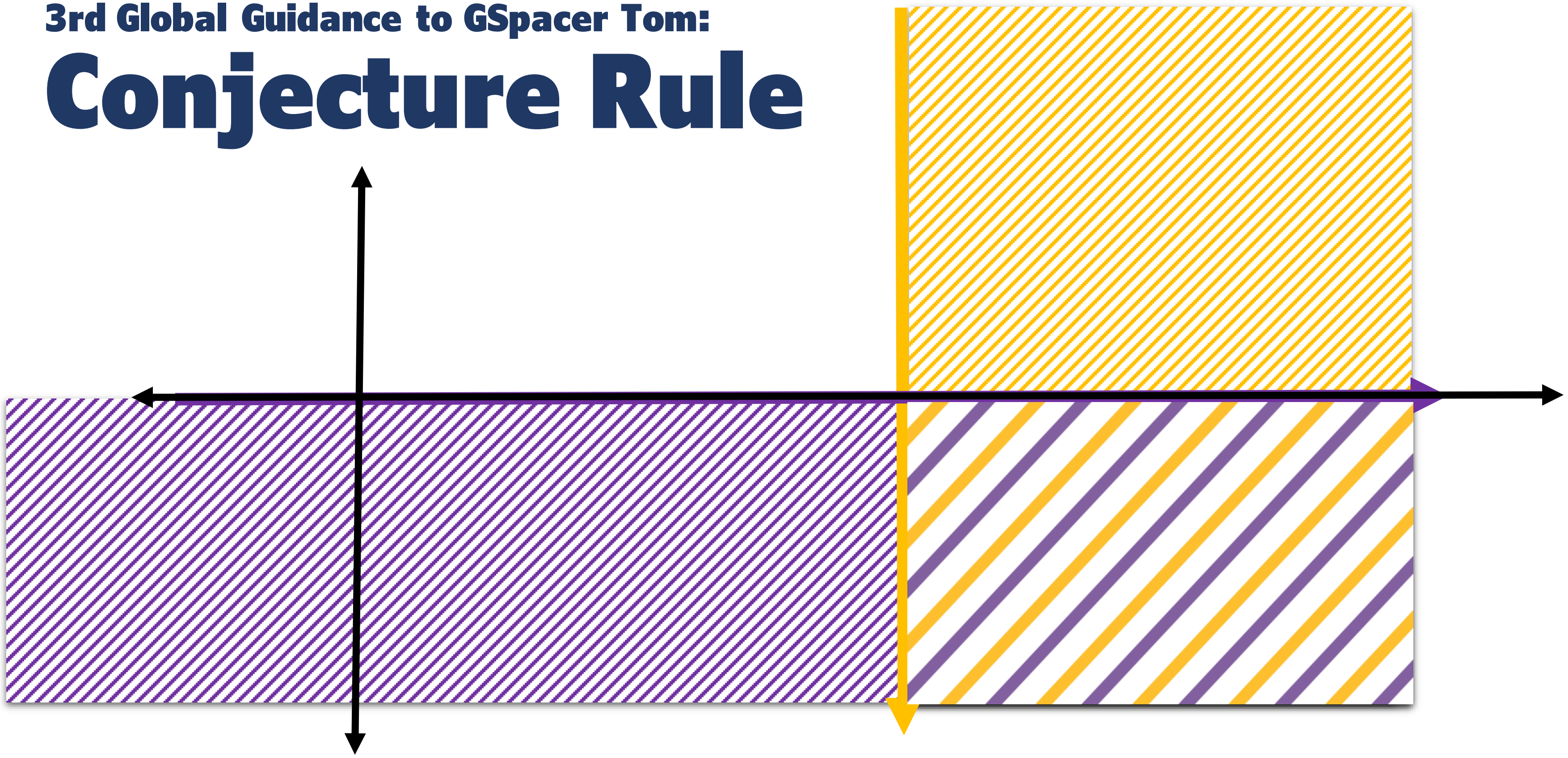
Concretize Rule



```
if $(\forall \ell \in \mathcal{L} \cdot \ell$ partially blocks $\varphi) \wedge$
   $(\exists \gamma \cdot \gamma \Rightarrow \varphi \wedge (\gamma$ is not blocked by $\bigwedge \mathcal{L}))$ then
     add $\gamma$ to POB queue
```

3rd Global Guidance to GSpacer Tom:

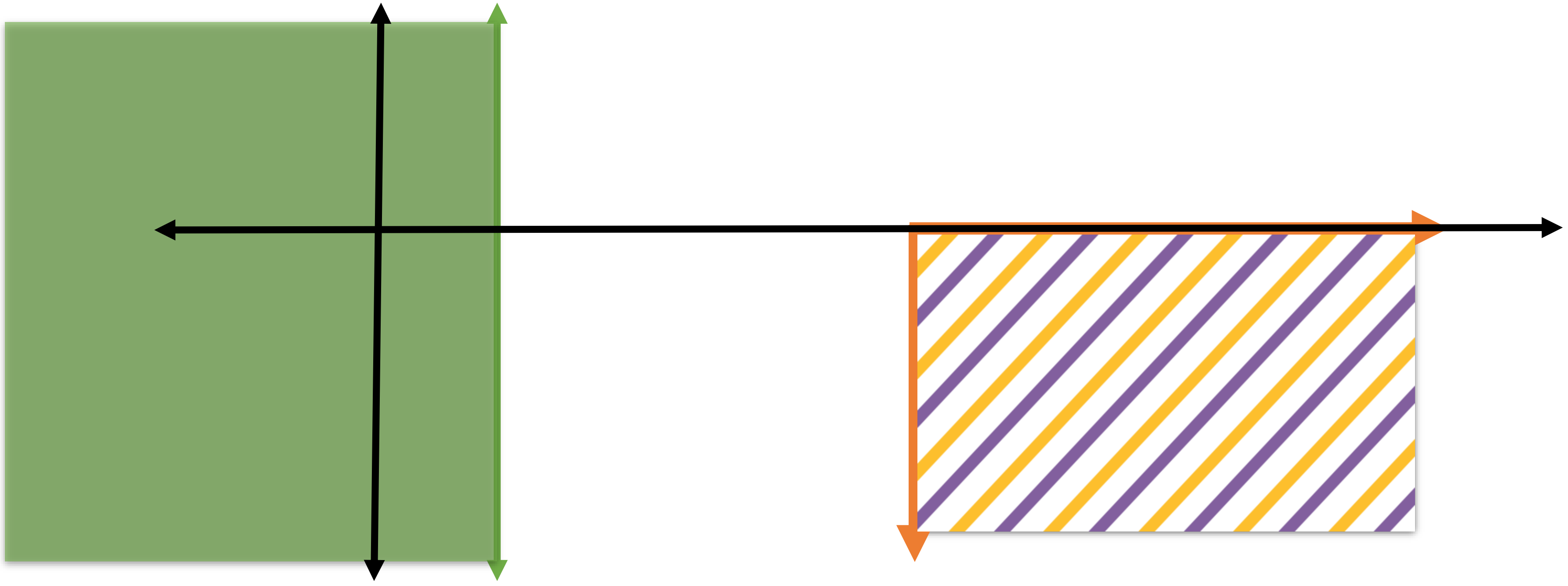
Conjecture Rule



```
if ($\varphi \equiv \alpha \wedge \beta$) ∧
    ($\forall \ell \in \mathcal{L} \cdot \ell$ blocks $\beta$ but does not block $\alpha$) then
        add $\alpha$ to POB queue
```

3rd Global Guidance to GSpacer Tom:

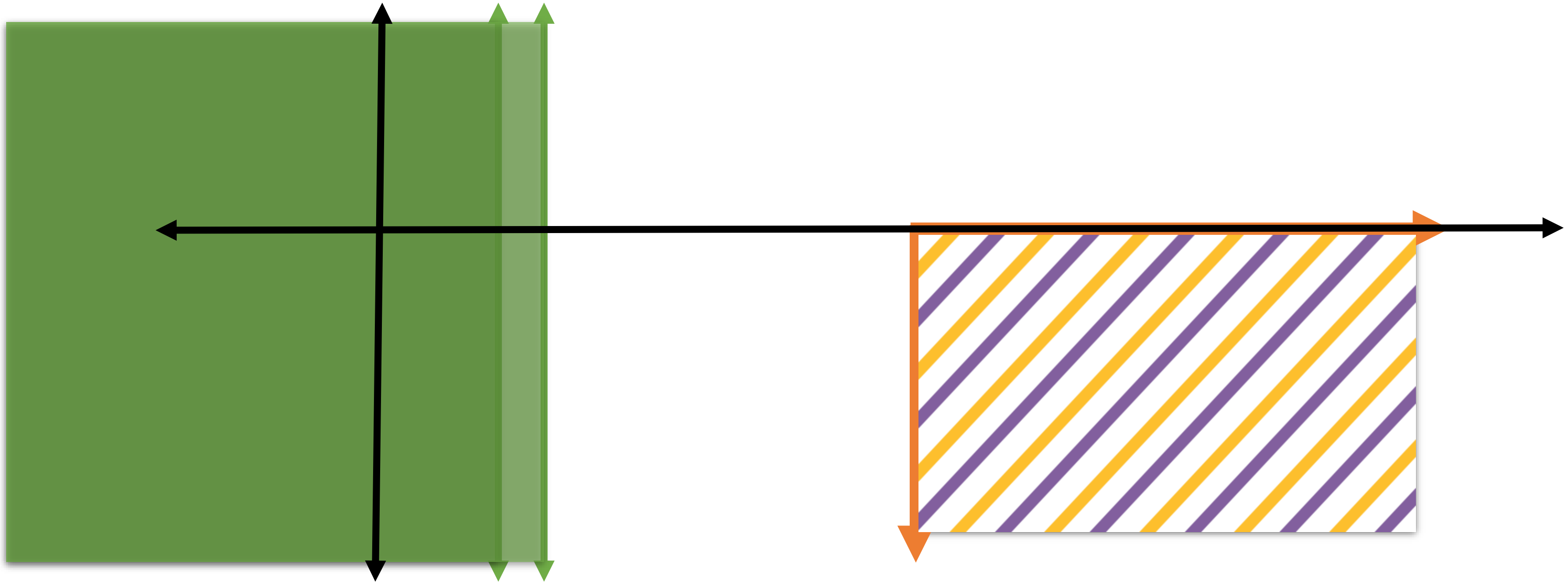
Conjecture Rule



```
if ($\varphi \equiv \alpha \land \beta$) $\land$
    ($\forall \ell \in \mathcal{L} \cdot \ell$ blocks $\beta$ but does not block $\alpha$) then
        add $\alpha$ to POB queue
```

3rd Global Guidance to GSpacer Tom:

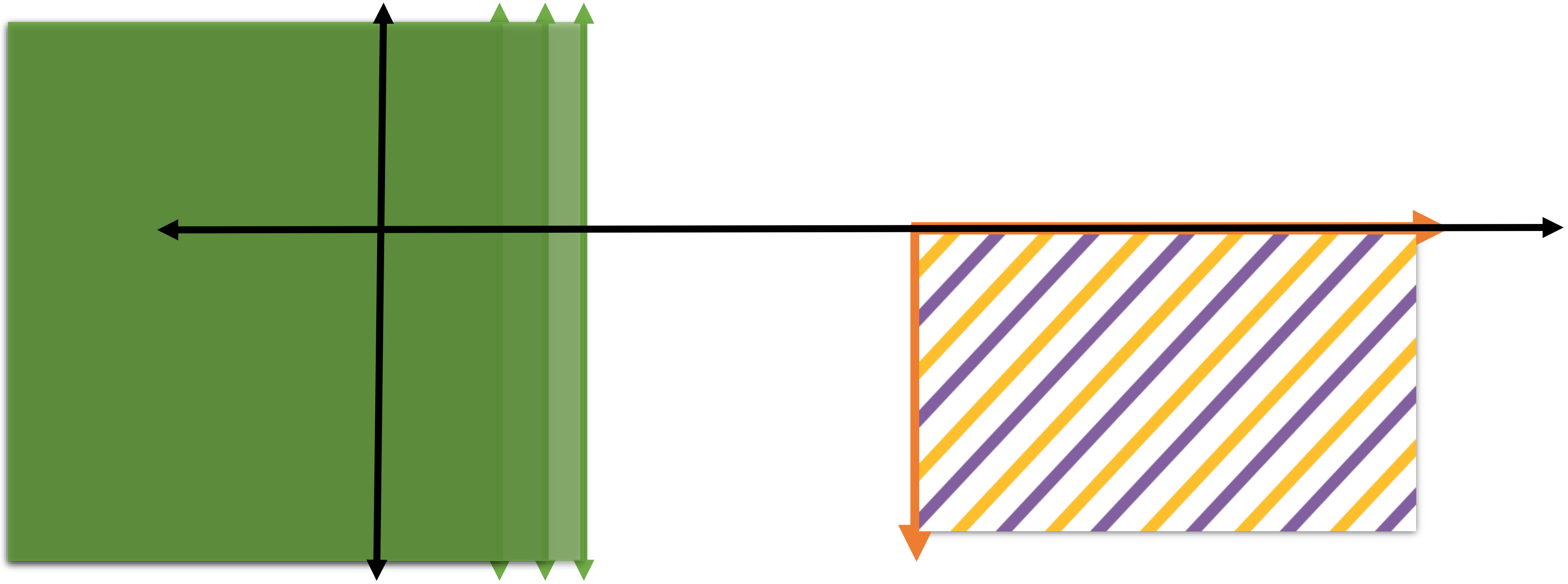
Conjecture Rule



```
if ($\varphi \equiv \alpha \wedge \beta$) ∧
    ($\forall \ell \in \mathcal{L} \cdot \ell$ blocks $\beta$ but does not block $\alpha$) then
        add $\alpha$ to POB queue
```

3rd Global Guidance to GSpacer Tom:

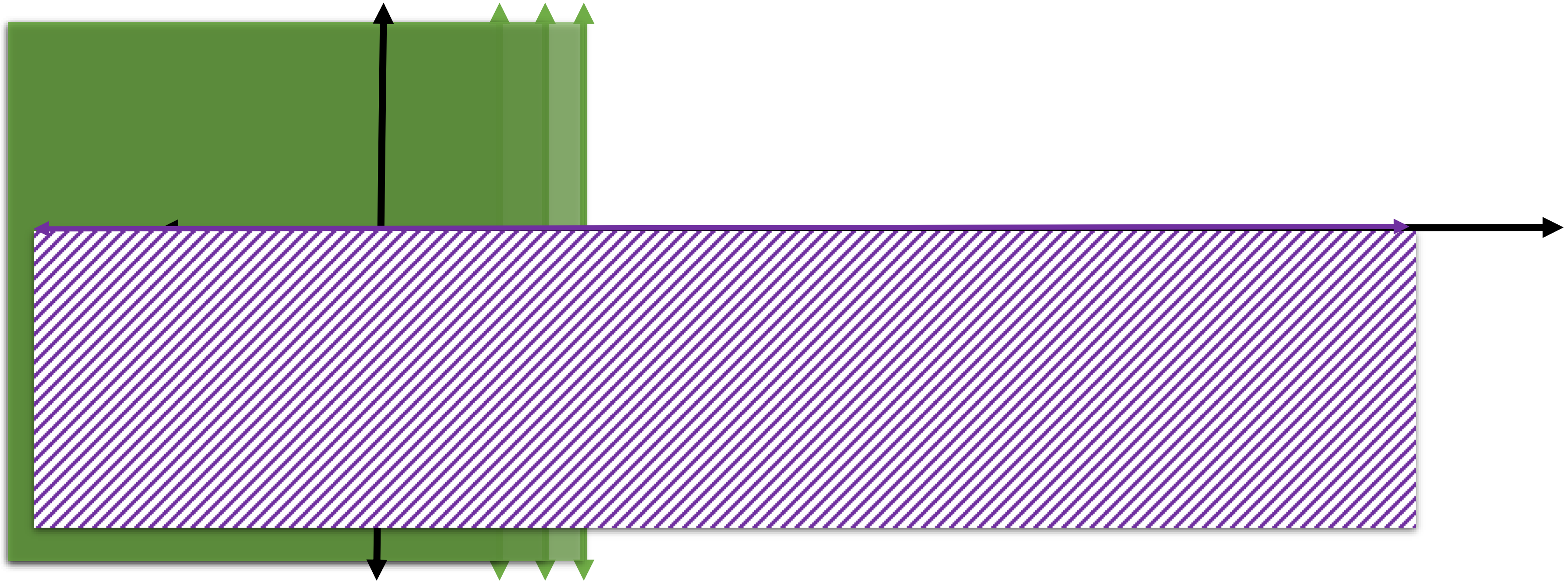
Conjecture Rule



```
if ($\varphi \equiv \alpha \wedge \beta$) ∧
   ($\forall \ell \in \mathcal{L} \cdot \ell$ blocks $\beta$ but does not block $\alpha$) then
      add $\alpha$ to POB queue
```

3rd Global Guidance to GSpacer Tom:

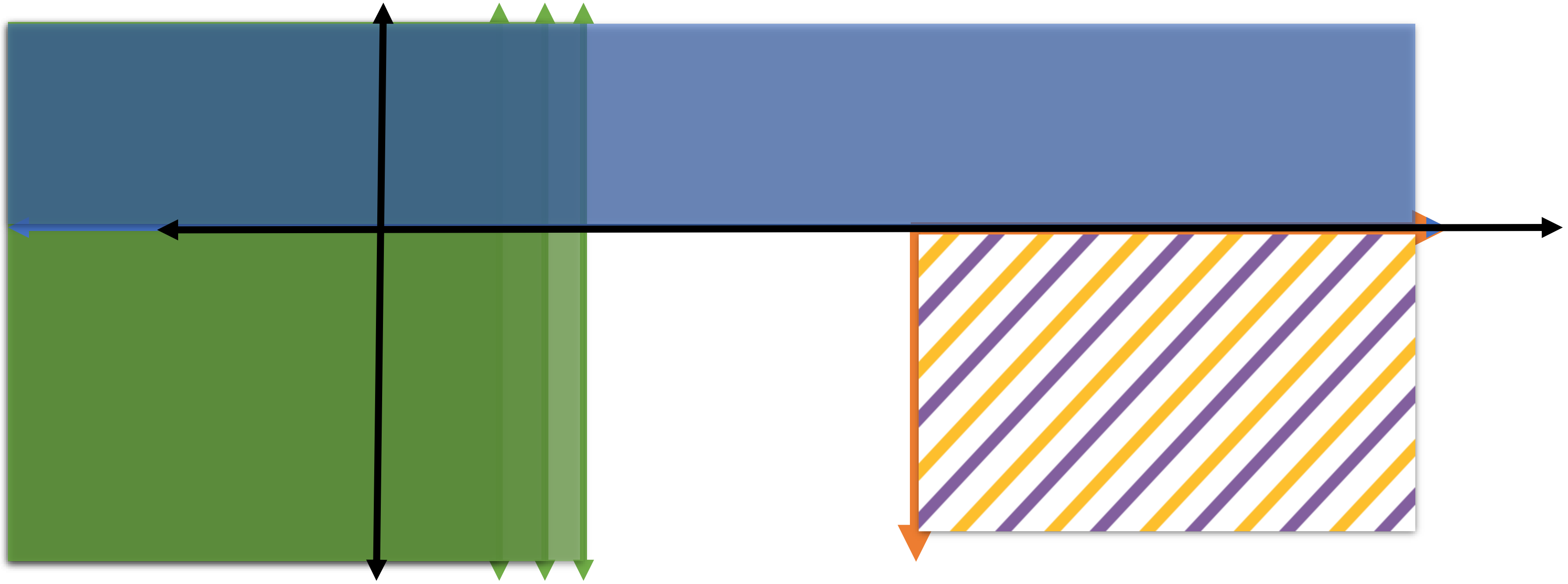
Conjecture Rule



```
if ($\varphi \equiv \alpha \wedge \beta$) ∧
   ($\forall \ell \in \mathcal{L} \cdot \ell$ blocks $\beta$ but does not block $\alpha$) then
      add $\alpha$ to POB queue
```

3rd Global Guidance to GSpacer Tom:

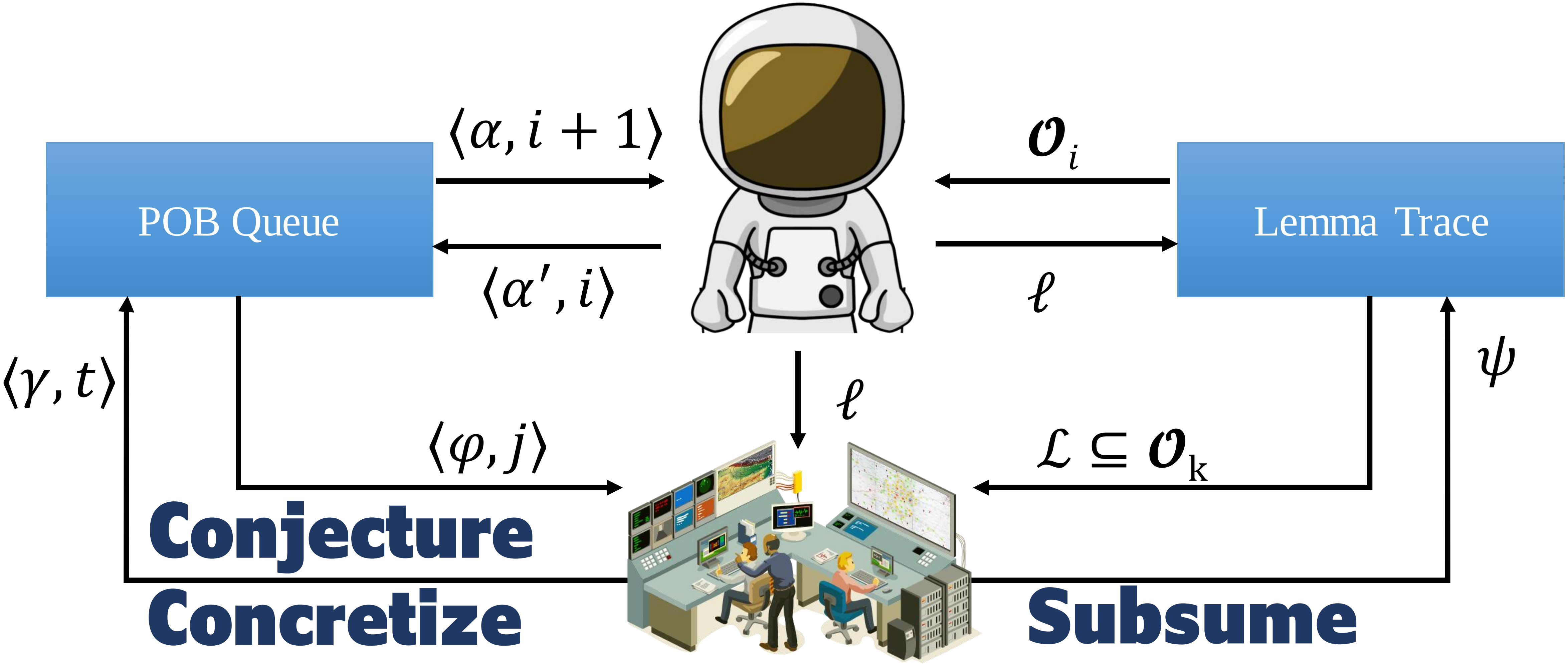
Conjecture Rule



```
if ($\varphi \equiv \alpha \wedge \beta$) ∧
    ($\forall \ell \in \mathcal{L} \cdot \ell$ blocks $\beta$ but does not block $\alpha$) then
        add $\alpha$ to POB queue
```

Ground Control to Spacer Tom:

Global Guidance



POB Queue

$\langle \alpha, i+1 \rangle$

$\langle \alpha', i \rangle$

$\mathcal{O}_i$

Lemma Trace

ℓ

$\langle \gamma, t \rangle$

ψ

ℓ

$\langle \varphi, j \rangle$

$\mathcal{L} \subseteq \mathcal{O}_{\mathrm{k}}$

Conjecture

Concretize

Subsume

Implementation and Evaluation

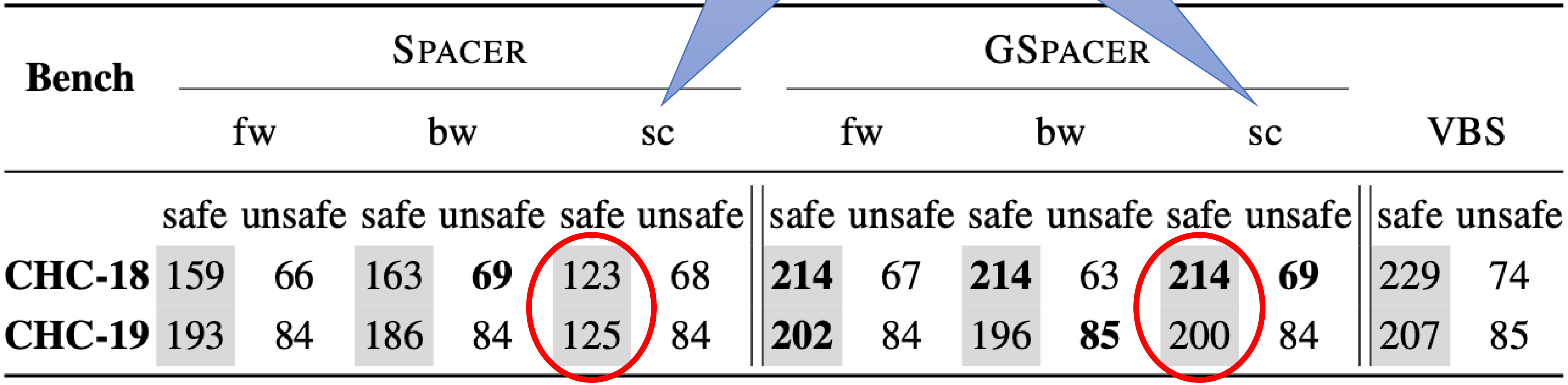
- As an extension to Spacer
  https://github.com/hgvk94/z3/tree/gspacer-cav-ae

- Supports
  - Linear Integer Arithmetic, Linear Real Arithmetic
  - Linear and Non-linear CHCs
  - Arrays and Fixed-Size Bit-Vectors ongoing

- Evaluated on LIA instances from CHC-COMP

Results

No interpolation!

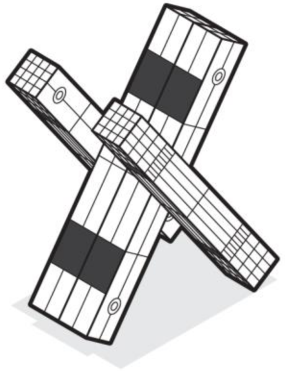


Bench	SPACER						GSPACER							
	fw		bw		sc		fw		bw		sc		VBS	
	safe	unsafe	safe	unsafe	safe	unsafe	safe	unsafe	safe	unsafe	safe	unsafe	safe	unsafe
CHC-18	159	66	163	**69**	123	68	**214**	67	**214**	63	**214**	**69**	229	74
CHC-19	193	84	186	84	125	84	**202**	84	196	**85**	200	84	207	85

fw and ***bw*** are different interpolation strategies.
sc configuration disables interpolation.

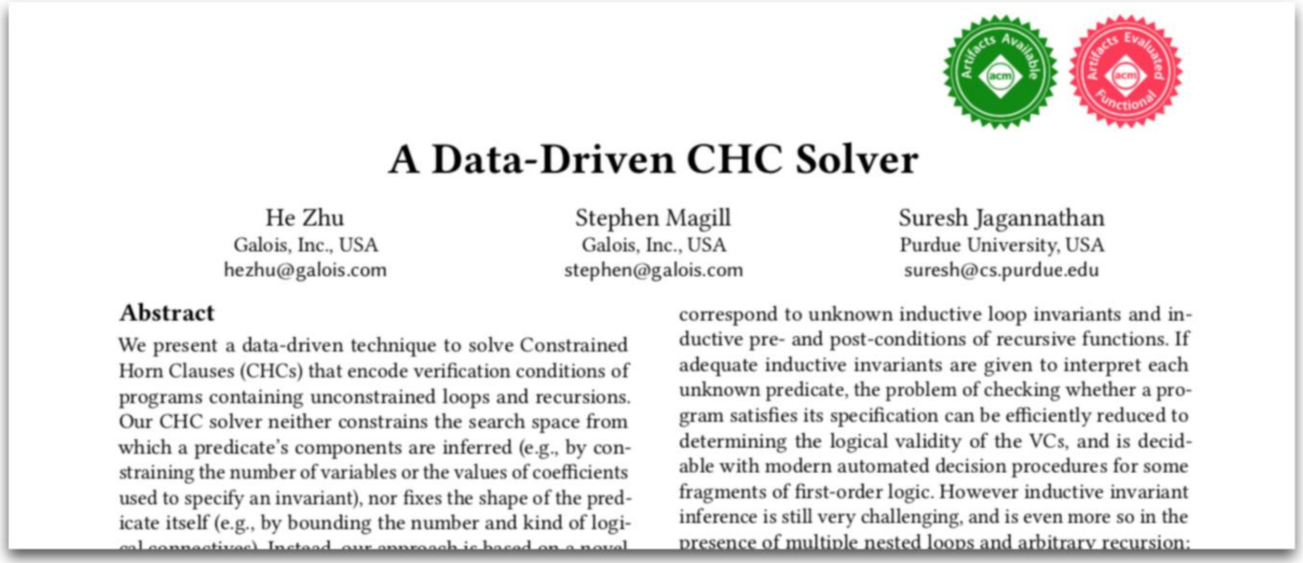
GSpacer won 3 of the 4 tracks at CHC-COMP 2020

Linear Arbitrary (LArb) from PLDI 18



Data-driven, machine learning based invariant inference algorithm

Evaluation showed promise on a subset of SV-COMP benchmarks



A Data-Driven CHC Solver

He Zhu
Galois, Inc., USA
hezhu@galois.com

Stephen Magill
Galois, Inc., USA
stephen@galois.com

Suresh Jagannathan
Purdue University, USA
suresh@cs.purdue.edu

Abstract

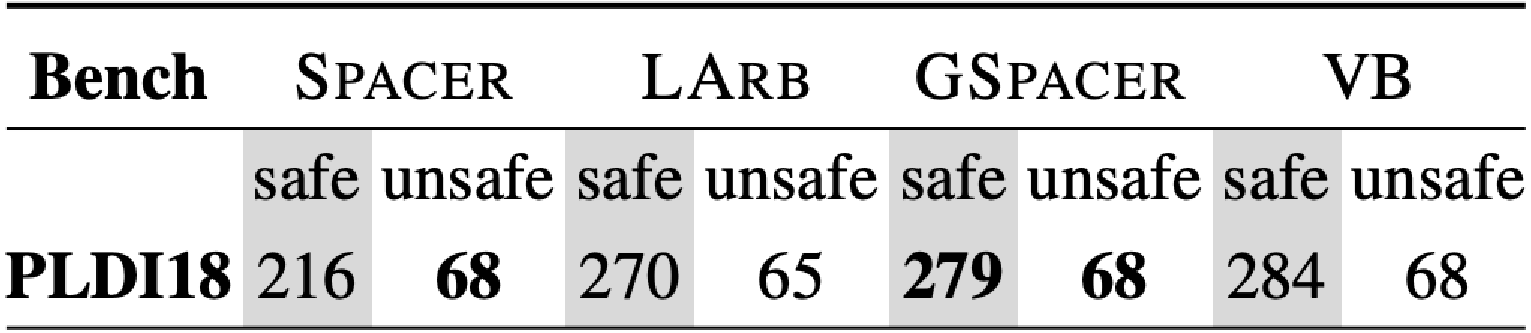
We present a data-driven technique to solve Constrained Horn Clauses (CHCs) that encode verification conditions of programs containing unconstrained loops and recursions. Our CHC solver neither constrains the search space from which a predicate's components are inferred (e.g., by constraining the number of variables or the values of coefficients used to specify an invariant), nor fixes the shape of the predicate itself (e.g., by bounding the number and kind of logical connectives). Instead, our approach is based on a novel

correspond to unknown inductive loop invariants and inductive pre- and post-conditions of recursive functions. If adequate inductive invariants are given to interpret each unknown predicate, the problem of checking whether a program satisfies its specification can be efficiently reduced to determining the logical validity of the VCs, and is decidable with modern automated decision procedures for some fragments of first-order logic. However inductive invariant inference is still very challenging, and is even more so in the presence of multiple nested loops and arbitrary recursion;

We compared GSpacer with LArb

- Could not compare on CHC-COMP instances as LArb solved significantly fewer instances than even Spacer

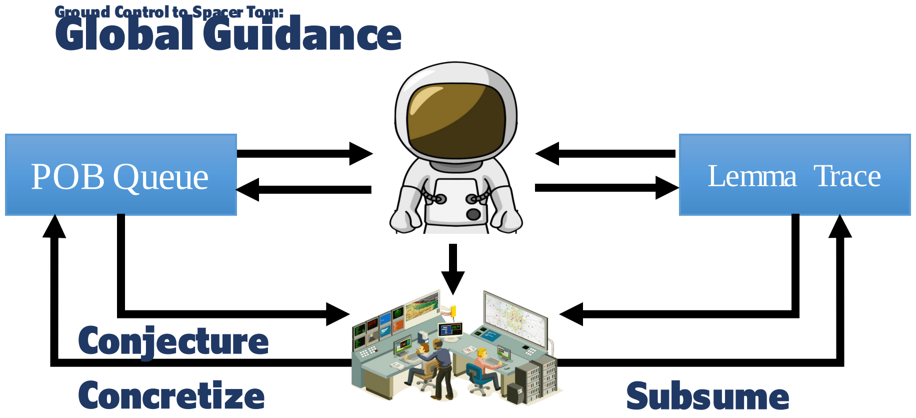
- Compared **on benchmarks from LArb paper**

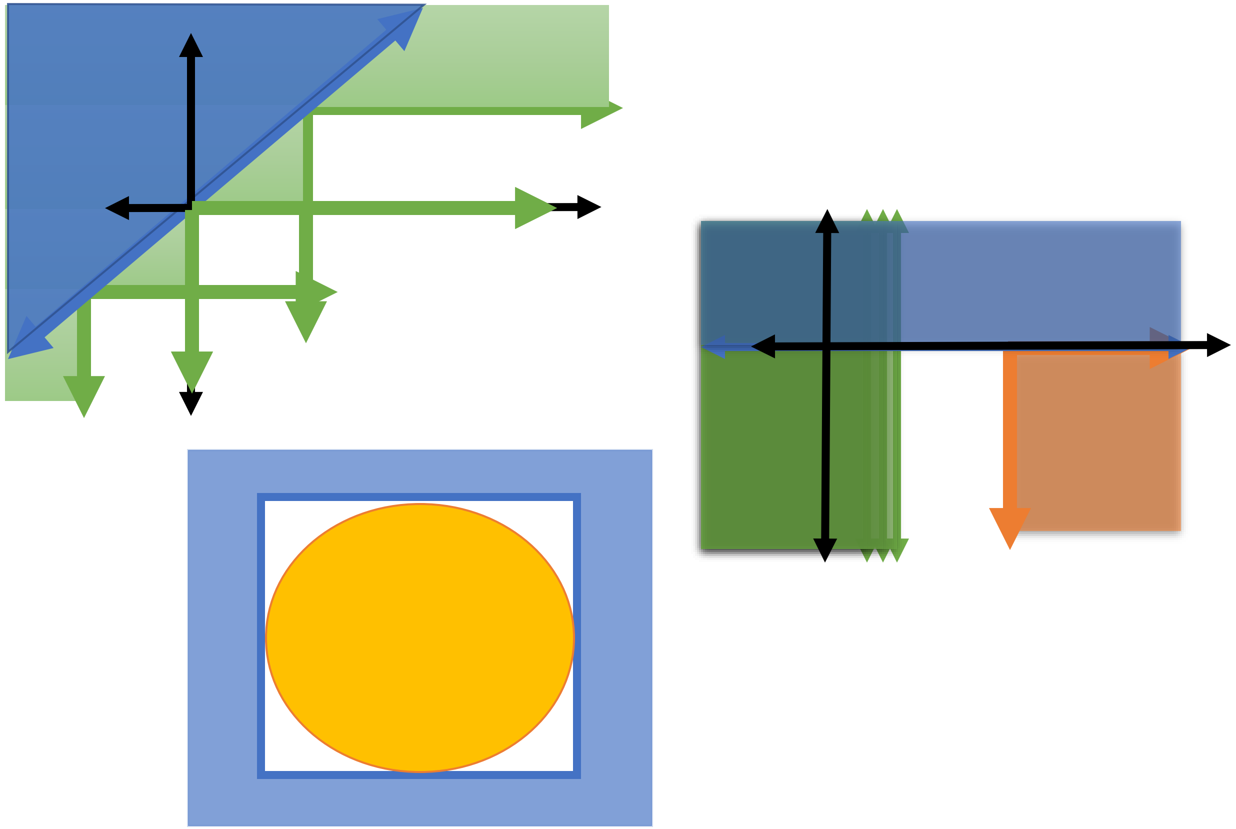


Bench	Spacer		LArb		GSpacer		VB	
	safe	unsafe	safe	unsafe	safe	unsafe	safe	unsafe
PLDI18	216	**68**	270	65	**279**	**68**	284	68

VB stands for virtual best

Conclusion





- Global guidance technique to mitigate limitations of local reasoning
- Stable under different interpolation strategies
- Data driven guidance for MC is better than both invariant inference and local reasoning

Future Work

- Extend to theories where there is no interpolation
  - BV, Arrays

- Add more rules
  - Symmetry breaking in distributed protocol verification

Thanks for listening

https://hgvk94.github.io/gspacer/