Abstract Domains for Automated Reasoning about List-Manipulating Programs with Infinite Data

Constantin Enea
University Paris Diderot (Paris 7)

Joint work with

Ahmed Bouajjani
U Paris 7

Cezara Dragoi
IST Austria

Mihaela Sighireanu
U Paris 7

CP meets CAV, June 2012 -- Turkey

This Talk?

Automated Verification of Partial Correctness

Sequential Programs with

- Dynamic singly-linked lists
- Infinite data (Integers)

Based on [CAV'10] , [PLDI'11], [VMCAI'12], [ATVA'12]

Goal

"head points to a singly-linked list"

```
void firstEven(list* head){
    list* h=head;
    int k = 0;
    while(h!=NULL){
          h->data =2*k;
          k++;
          h=h->next;
     }
```

"head points to a singly-linked list containing the **first even numbers**"

Goal

"head points to a singly-linked list"

```
list* quicksort(list* head) {
    list *left,*right,* a=head;
    if (a=NULL) return a;
          list* pivot=create();
    pivot->data =a->data;
    split(a->next,left,right,pivot->data);

    left = quicksort(left);
    right = quicksort(right);

    ret = concat(left,pivot,right);
    return ret; }
```

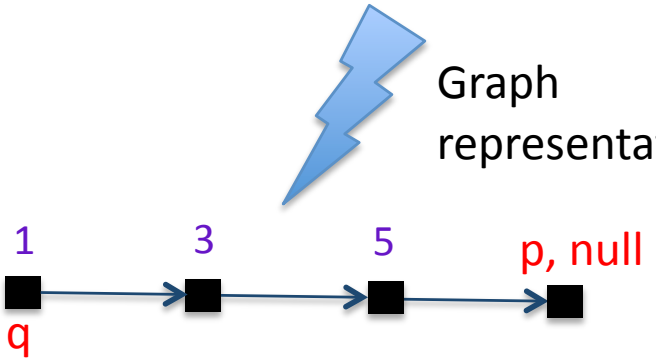
"ret points to a **sorted** singly-linked list"

Goal

```
void main(){
    list* h1,h2;
    h1=init(N);
    h2=copy(h1);

    assert("h1 and h2 point to two  equal lists");

    h1=quicksort(h1);
    h2=mergesort(h2);

    assert("h1 and h2 point to two  equal lists");
}
```

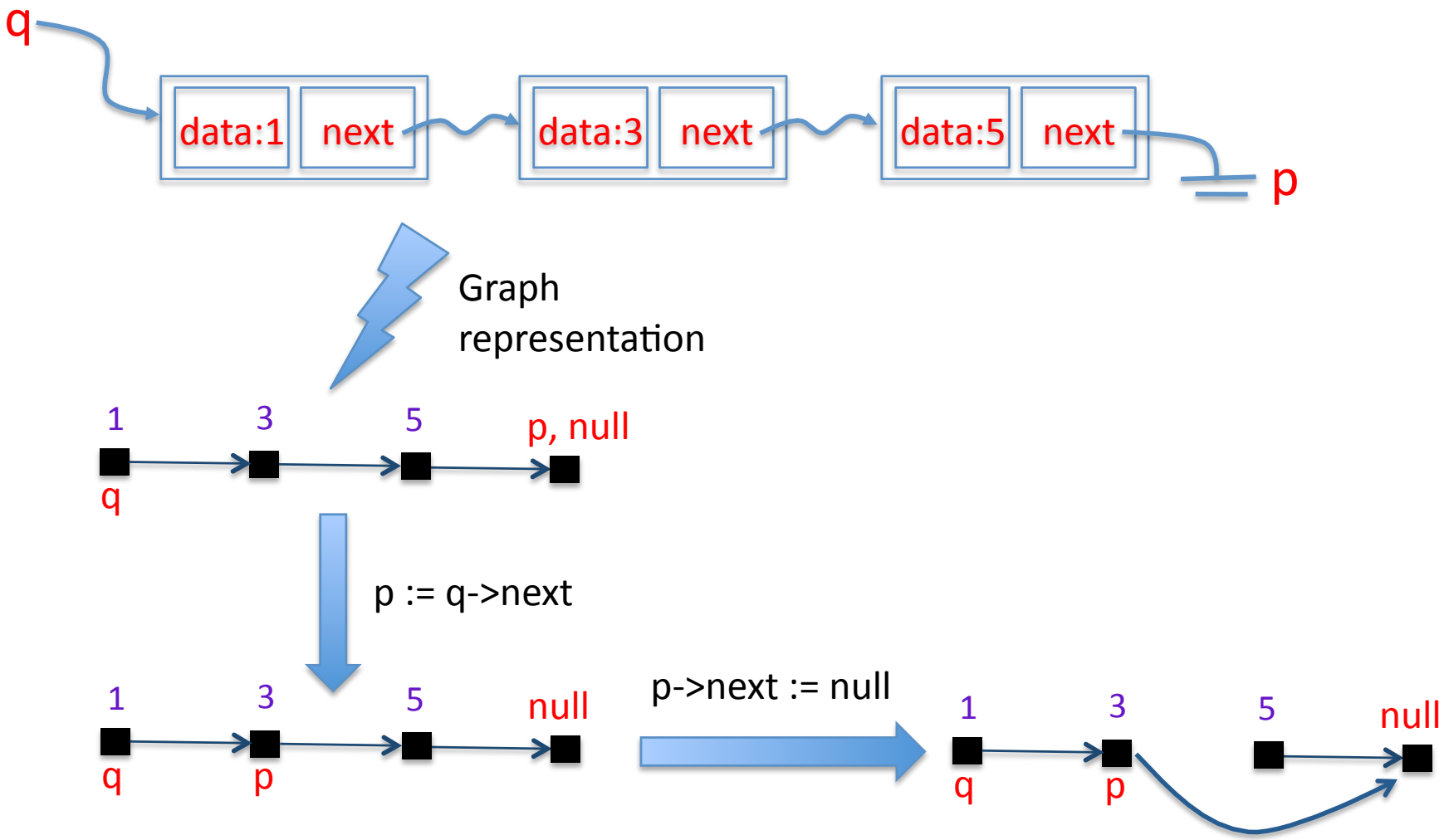
Program model

Transition system over graphs



Program model

Transition system over graphs



What kind of Specifications?

- Shape: *Input and Output are acyclic*
  - ➔ *FO + Reachability predicates*
- Sizes: *length(Input) = length(Output)*
  - ➔ *Arithmetical constraints*
- Multisets: *mset(Input) = mset(Output)*
  - ➔ *Multiset constraints*
- Data: *Output is sorted*
  - ➔ *FO + data constraints*

Pre/Post-condition reasoning

Given:
Annotated Program

```
@precondition  Φ_Pre
void proc ( … ){
    …
    while( … ){
     @loop invariant: Φ_Inv
          …
    }
}
@postcondition Φ_Post
```

Prove automatically that:

$$Post(\Phi, st) \Rightarrow \Phi'$$

➔ Issues:

- Expressiveness
- Closure under Post-image
- Checking Implication/SAT

Decidable Logics

- Logics for structure properties:
  - Mona+Graph types [Klarlund, Moeller '90's]
  - Separation logic [Berdine et al. '04]
  - LRP [Yorsh et al.'07]
- Logics for structure properties and data
  - LISBQ [Lahiri , Qadeer '08]
  - CSL [B., Dragoi, Enea, Sighireanu '09]
  - ADT+recursive functions [Suter, Dotta, Kuncak '10]
  - STRAND [Madhusudan, Parlato, Qiu'11]
- Logics on arrays and data:
  - APF [Bradley, Manna, Simpa' 06]
  - LIA [Habermehl, Iosif, Vojnar'08]

Decidable Logics: Limitations

- Difficult to reason about shapes
  - Reachability ?
- Difficult to combine decidable theories:
  - Structure + data constraints,
  - Heterogeneous data constraints
- Complexity

Annotations?

- Synthesis of Loop Invariants
- Synthesis of Procedure Specifications

- Invariants ≈ Reachable configurations
- Input/Output Relations ≈ Proc. Summaries

An AI-Based Approach

- Properties → Objects in an Abstract Domain
- Approximate operations:
  - Disjunction ≈ Meet
  - Conjunction ≈ Join
  - Implication ≈ Entailment
- Invariant Synthesis, procedure summaries → Modular Inter-procedural analysis

Related work

Shape analysis:

[Sagiv et al.'02, Rinetzky et al. '05],
[Chang et al. 08], [Calcagno et al. '09]
[Wies et al. '10], [Rival et al. '11]

Shape + Data:

[Chan, Rival '08], [McKloskey et al. '10]

Size constraints: [Gulwani et al. '09]

Universal formulas: [Gulwani et al. '08],
[Halbwachs et al. '08]

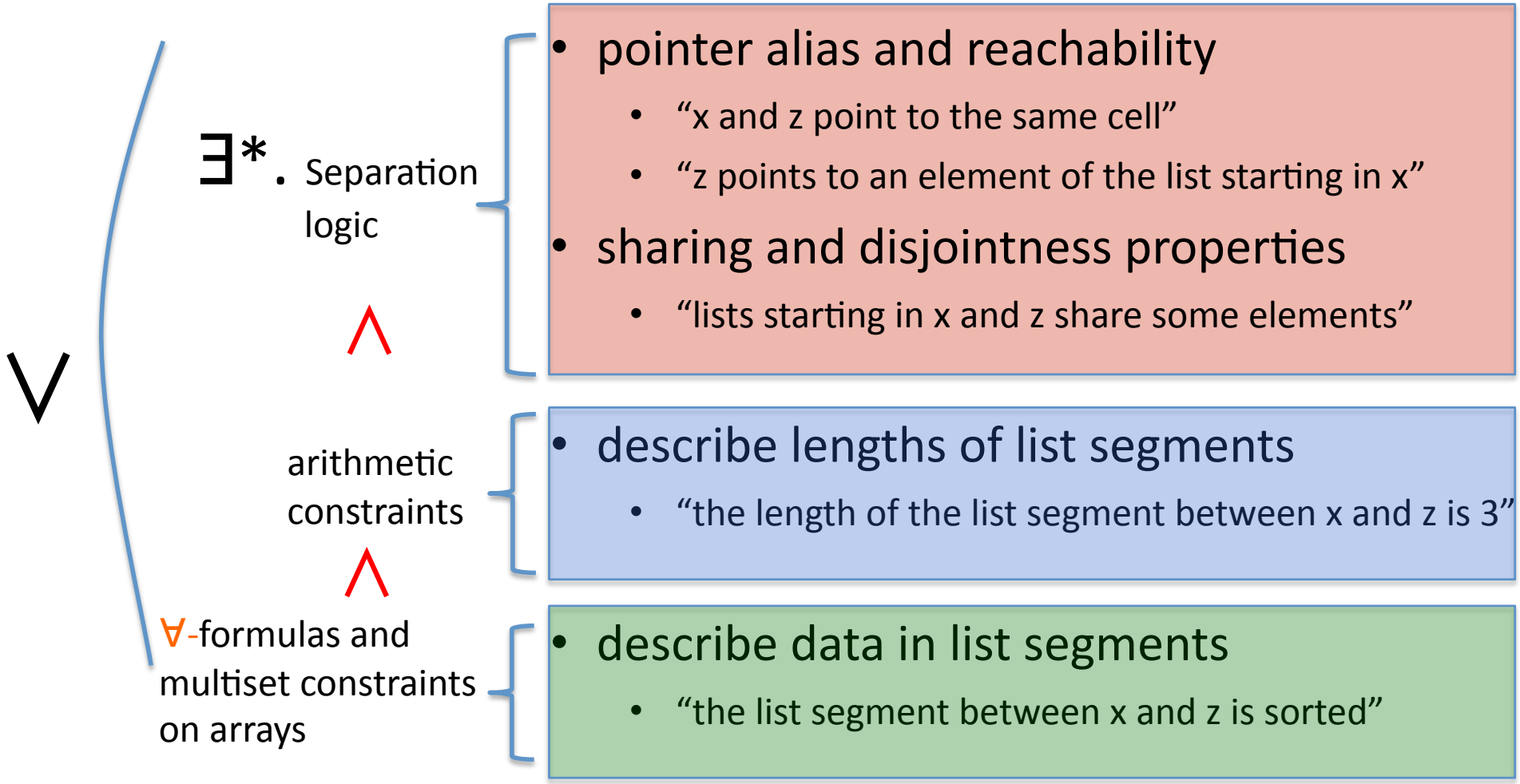
Multiset constraints: [Perrelle et al. '10]

Outline

- Logic SL3
- SL3: Entailment checking
- Abstract Domains for Heaps with Data
- Experimental results (CELIA)

Logic SL3

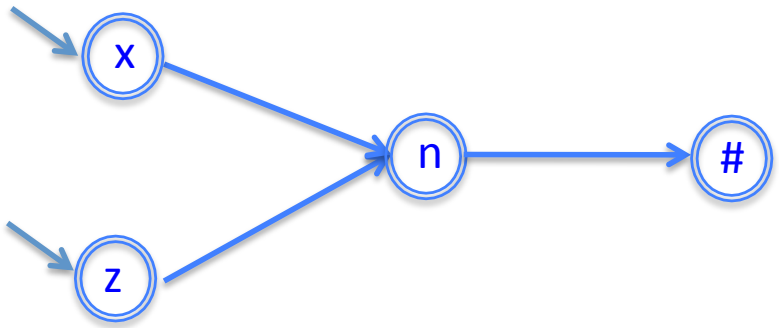
Systematic way of writing specifications



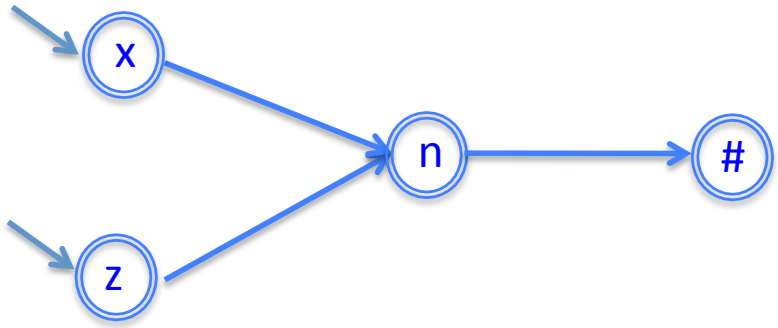
SL3: Shape constraints

“x and z point to two acyclic lists that share some elements”

$\exists$ n. ls(x,n) * ls(z,n) * ls(n,#)

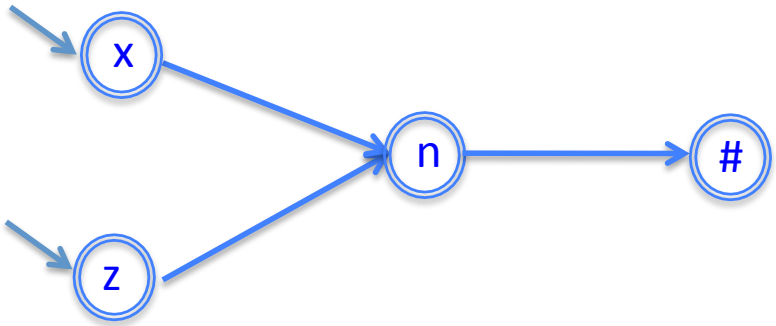


SL3: Length constraints



“the length of the list segment between x and n equals the length of the list segment between z and n”

SL3: Length constraints

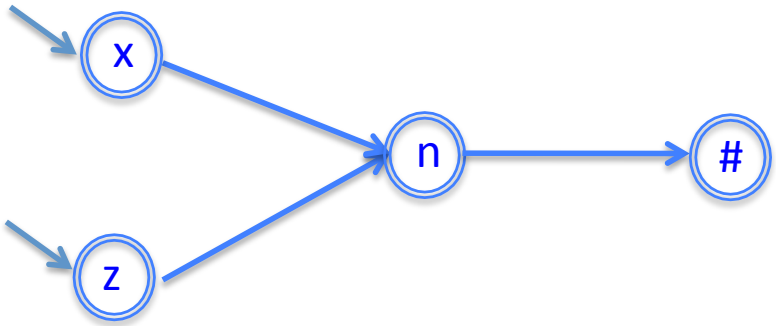


$\wedge$ len(x,n) = len(z,n)

"the length of the list segment between x and n equals the length of the list segment between z and n"

- in general, constraints in Presburger arithmetics

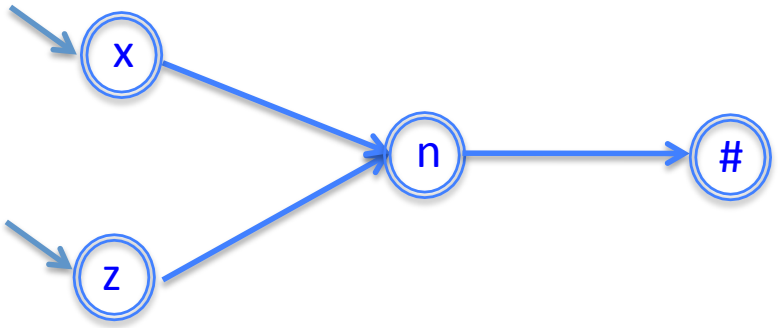
SL3: Data constraints



$\wedge$ len(x,n) = len(z,n)

“the list segment between x and n is sorted”

SL3: Data constraints



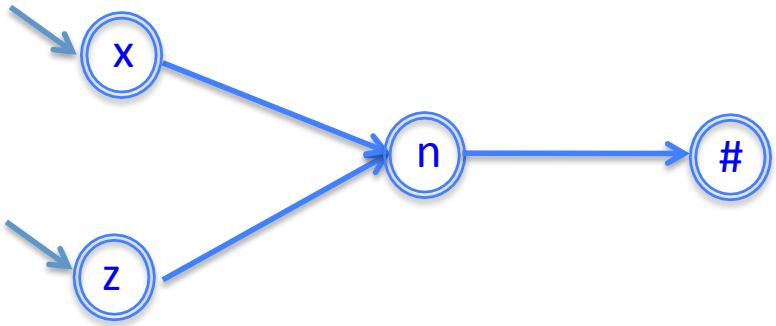
$\wedge$ len(x,n) = len(z,n)

$\wedge \quad \forall\, y1, y2 \in (x,n).\ y1 \leq y2 \Rightarrow (x,n)[y1] <= (x,n)[y2]$

"the list segment between x and n is sorted"

- in general, formulas of the form $\forall$ y. G(y) => U(y)

SL3: Data constraints

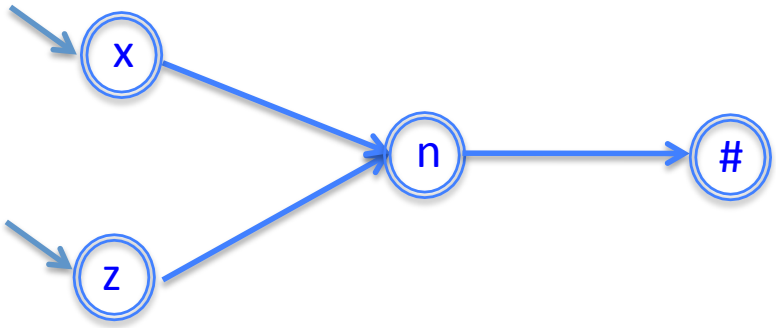


$\wedge$ len(x,n) = len(z,n)

$\wedge \quad \forall\, y1, y2 \in (x,n).\ y1 \leq y2 \Rightarrow (x,n)[y1] <= (x,n)[y2]$

"the multiset of elements in (x,n) equals the multiset of elements in (z,n)"

SL3: Data constraints



$\wedge$ len(x,n) = len(z,n)

$\wedge$ $\forall$ y1, y2 $\in$ (x,n). y1 $\leq$ y2 $\Rightarrow$ (x,n)[y1] <= (x,n)[y2]

$\wedge$ ms(x,n) = ms(z,n)

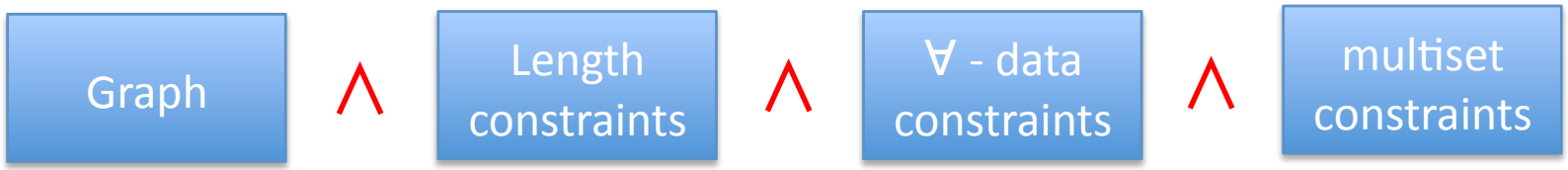
"the multiset of elements in (x,n) equals the multiset of elements in (z,n)"

- in general, equalities between unions of multisets

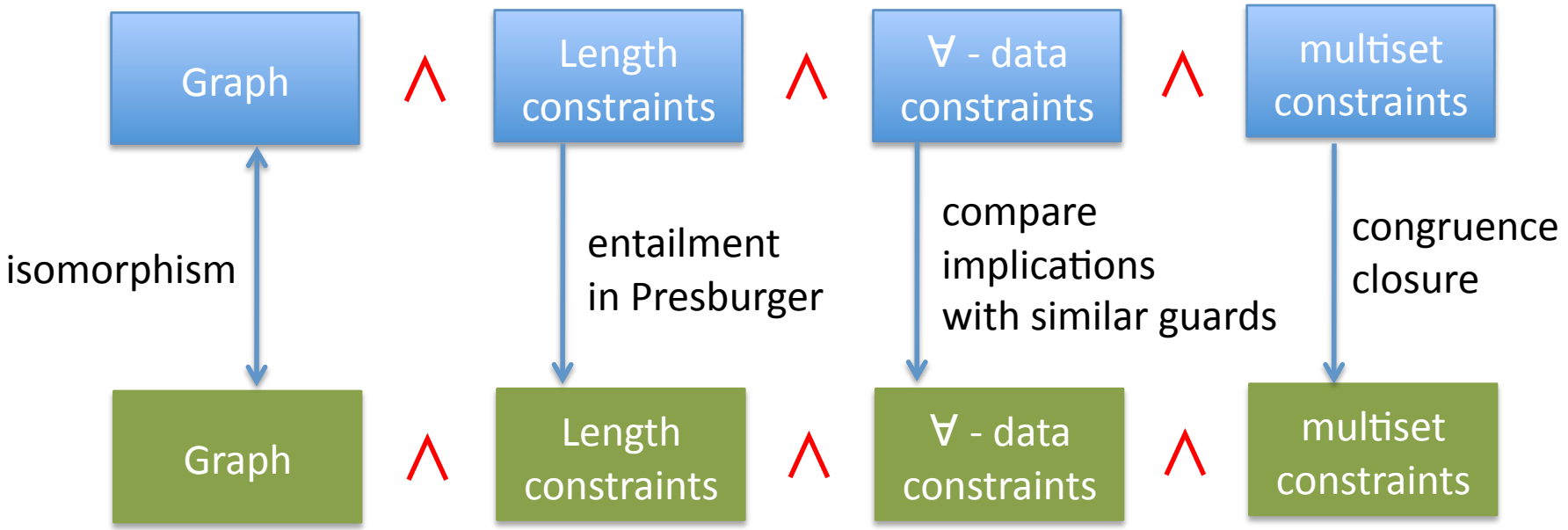
Outline

- Logic SL3
- SL3: Entailment checking
- Abstract Domains for Heaps with Data
- Experimental results (CELIA)

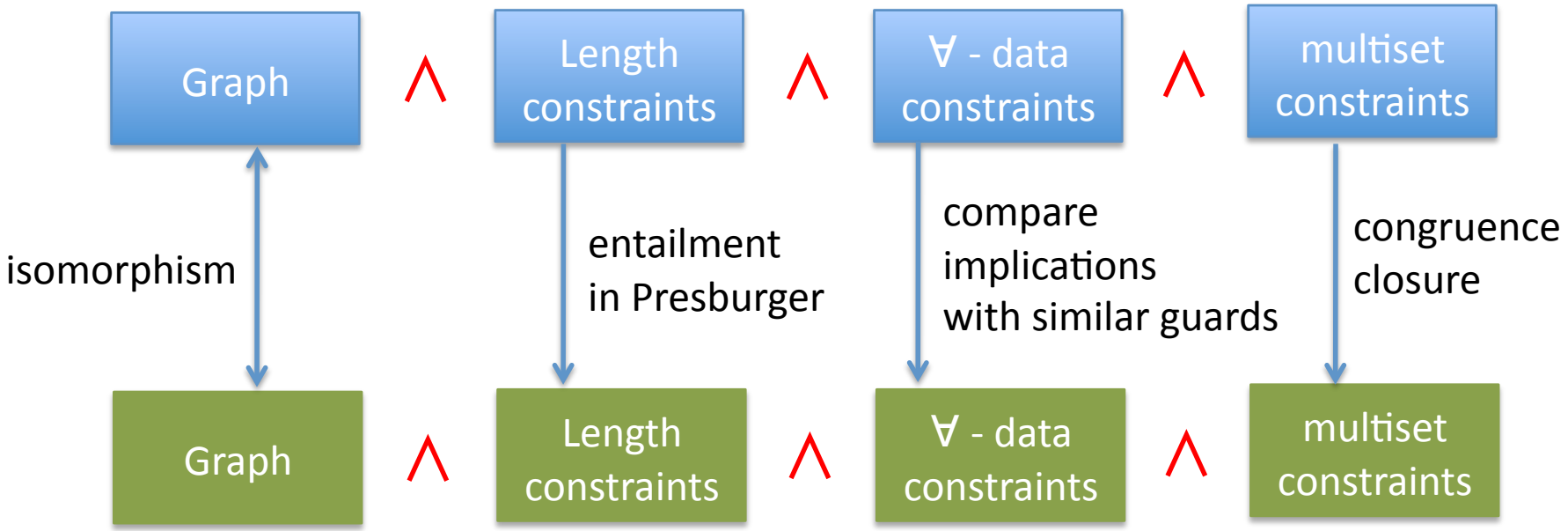
Sound entailment procedure



Sound entailment procedure



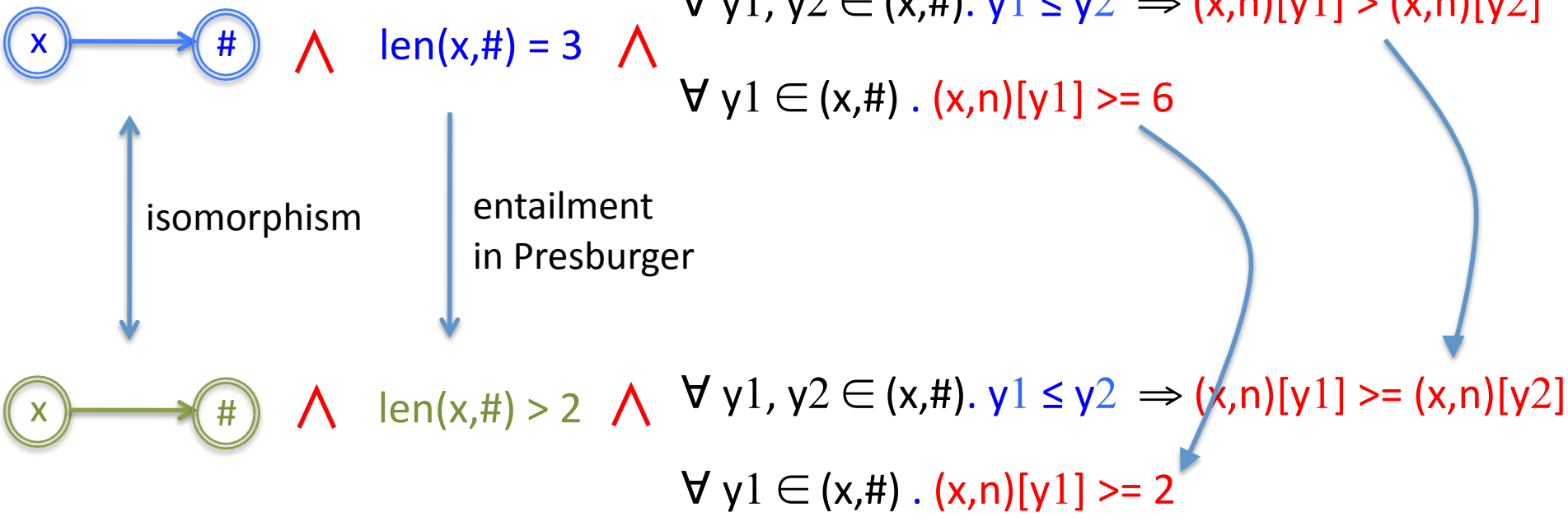
Sound entailment procedure



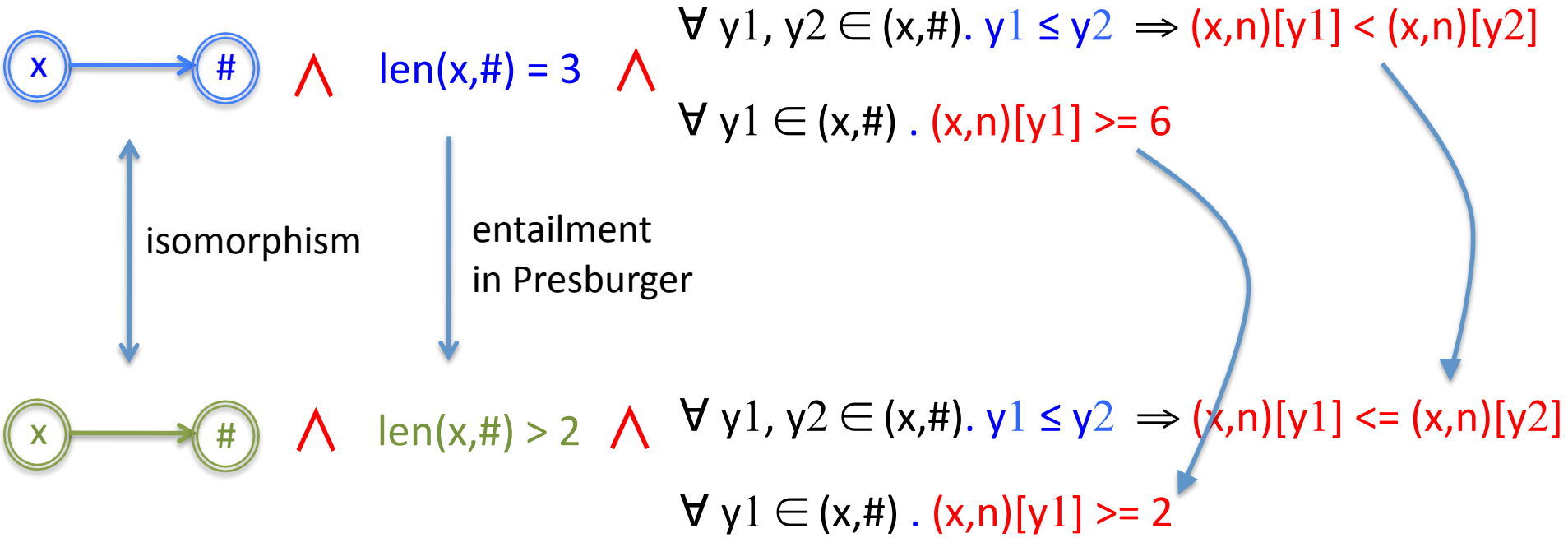
$$\bigvee_i \phi_i \Rightarrow \bigvee_j \phi_j$$

if for each ϕ_i there exists ϕ_j s.t. $\phi_i \Rightarrow \phi_j$

Sound entailment procedure



Sound entailment procedure



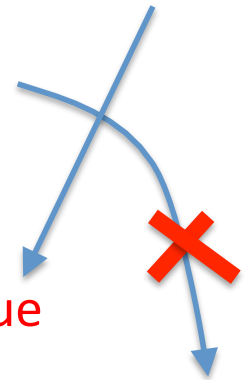
Improving precision:

- graph isomorphism -> graph homomorphism
- entailment of ∀-formulas -> propagate information between implications
- ∀-formulas + multiset constraints -> propagate information between the two types of constraints

Entailment of ∀-formulas

∀ y1, y2 ∈ (x,#). y1 ≤ y2 ⇒ (x,n)[y1] <= (x,n)[y2]

∀ y1 ∈ (x,#) . y2 = y1 + 1 ⇒ true



∀ y1, y2 ∈ (x,#). y1 ≤ y2 ⇒ true

∀∀ y1 ∈ (x,#) . y2 = y1 + 1 ⇒ (x,n)[y1] <= (x,n)[y2]

Entailment of ∀-formulas

$\forall\ y1, y2 \in (x,\#).\ y1 \leq y2 \Rightarrow (x,n)[y1] <= (x,n)[y2]$

$\forall\ y1 \in (x,\#)\ .\ y2 = y1 + 1 \Rightarrow$ true

Propagation

$y2 = y1 + 1 \Rightarrow\ y1 \leq y2$

$\forall\ y1, y2 \in (x,\#).\ y1 \leq y2 \Rightarrow (x,n)[y1] <= (x,n)[y2]$

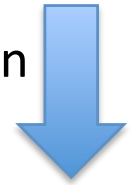
$\forall\ y1 \in (x,\#)\ .\ y2 = y1 + 1 \Rightarrow (x,n)[y1] <= (x,n)[y2]$

Entailment of ∀-formulas

$\forall\, y1, y2 \in (x,\#).\ y1 \le y2 \Rightarrow (x,n)[y1] <= (x,n)[y2]$

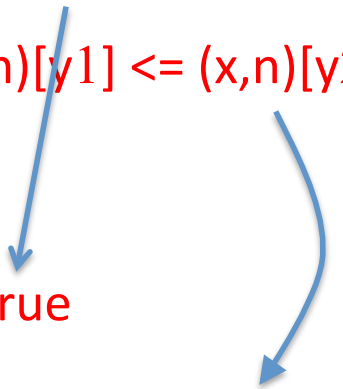
$\forall\, y1 \in (x,\#)\ .\ y2 = y1 + 1 \Rightarrow \text{true}$

Propagation



$\forall\, y1, y2 \in (x,\#).\ y1 \le y2 \Rightarrow (x,n)[y1] <= (x,n)[y2]$

$\forall\, y1 \in (x,\#)\ .\ y2 = y1 + 1 \Rightarrow (x,n)[y1] <= (x,n)[y2]$



$\forall\, y1, y2 \in (x,\#).\ y1 \le y2 \Rightarrow \text{true}$

$\forall\forall\, y1 \in (x,\#)\ .\ y2 = y1 + 1 \Rightarrow (x,n)[y1] <= (x,n)[y2]$

∀-formulas + multiset constr.

greater_than_5 **(x,#)**

ms **(x,#)** = ms **(y,#)**

=> greater_than_5 **(y,#)**

sortedness **(x,#)**

sortedness **(y,#)**

ms **(x,#)** = ms **(y,#)**

=> equality(**(x,#)** **(y,#)**)

Outline

- Logic SL3
- SL3: Entailment checking
- Abstract Domains for Heaps with Data
- Experimental results (CELIA)

Abstract interpretation

Goal: computed sound approximations of the set of reachable states

Concrete semantics:

$$X_0 = I$$
$$X_{i+1} = X_i \cup \text{post}(X_i)$$

Abstract domain = a class of constraints that form a lattice

- order relation -> an over-approximation of the entailment
- lub (join) and widening -> an over-approximation of disjunction
- glb (meet) -> an under-approximation of conjunction

Abstract semantics:

$$Y_0 = \alpha(I)$$
$$Y_{i+1} = Y_i \nabla \text{post}^{\#}(Y_i)$$

SL3 abstract domains

Parameters:

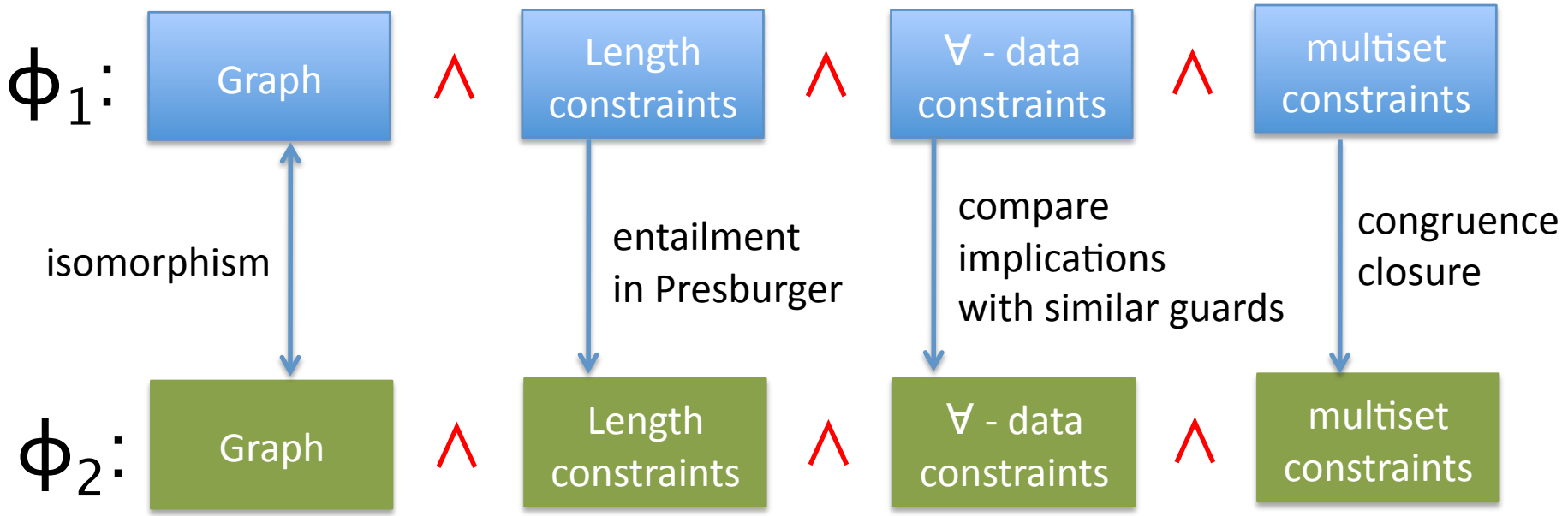
- a set of guards $\mathcal{G}$
- a numerical abstract domain $\mathcal{A}_Z$: intervals, octagons, polyhedra,..

SL3 abstract domain = SL3 formulas s.t. $\forall$-implications are of the form:

$\forall$ y. G(y) => U(y) with G(y) $\in \mathcal{G}$ and U(y) $\in \mathcal{A}_Z$

Order relation

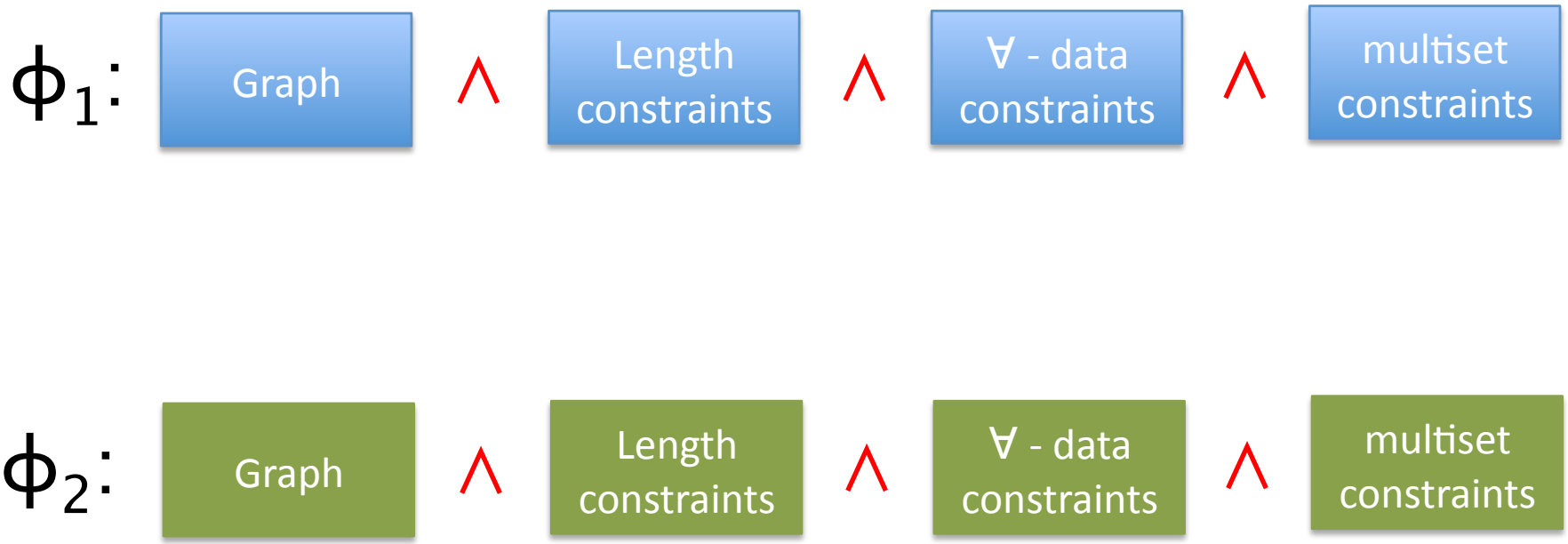
$\Phi_1 \sqsubseteq \Phi_2$ iff



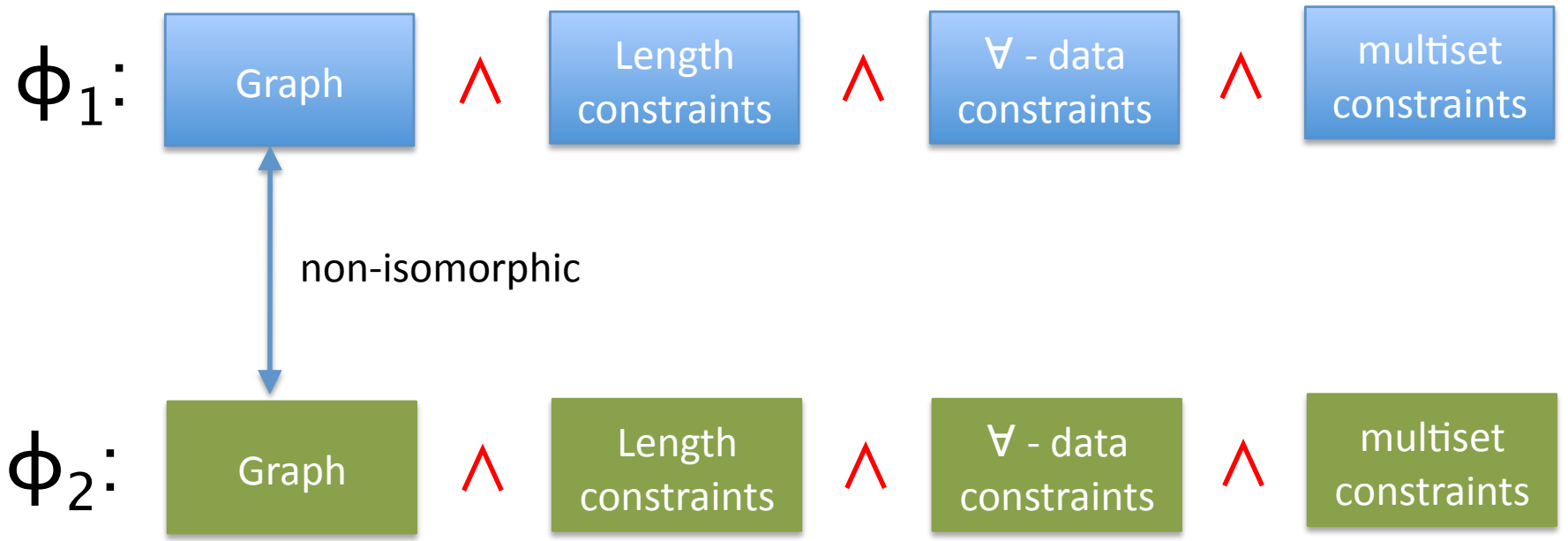
$\bigvee_i \Phi_i \sqsubseteq \bigvee_j \Phi_j$

iff for each Φ_i there exists Φ_j s.t. $\Phi_i \sqsubseteq \Phi_j$

Join operator

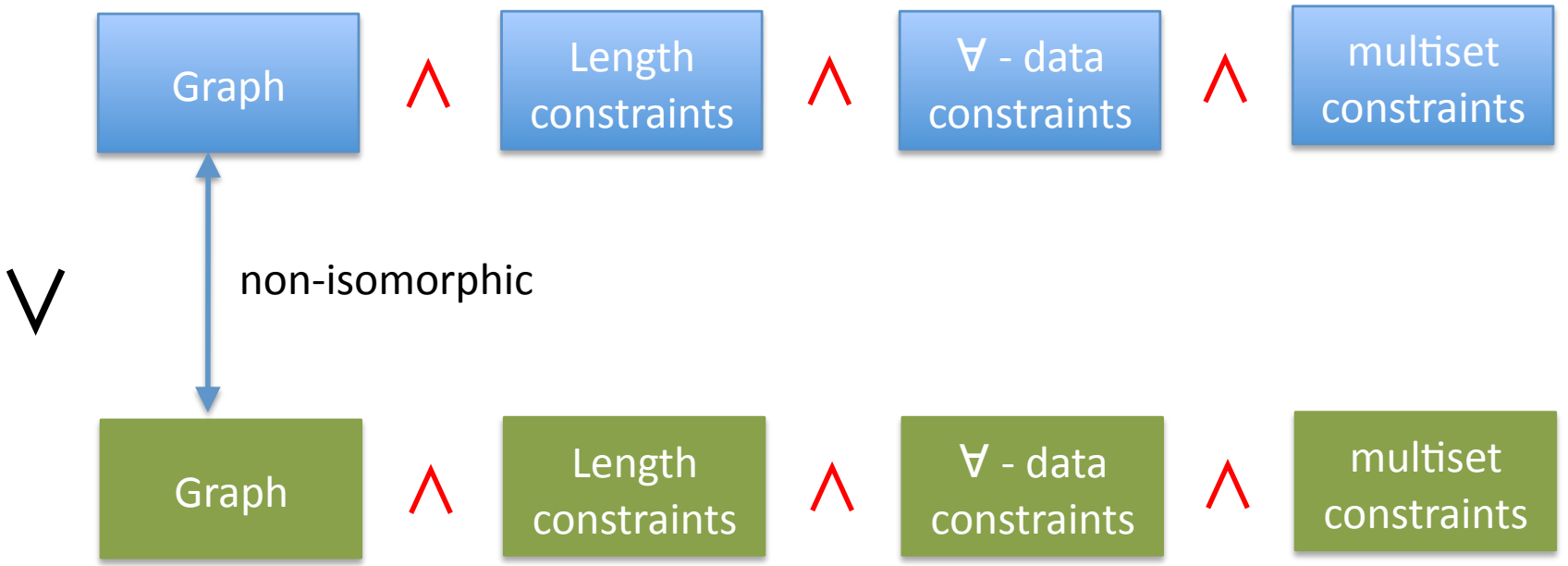


Join operator (case 1)

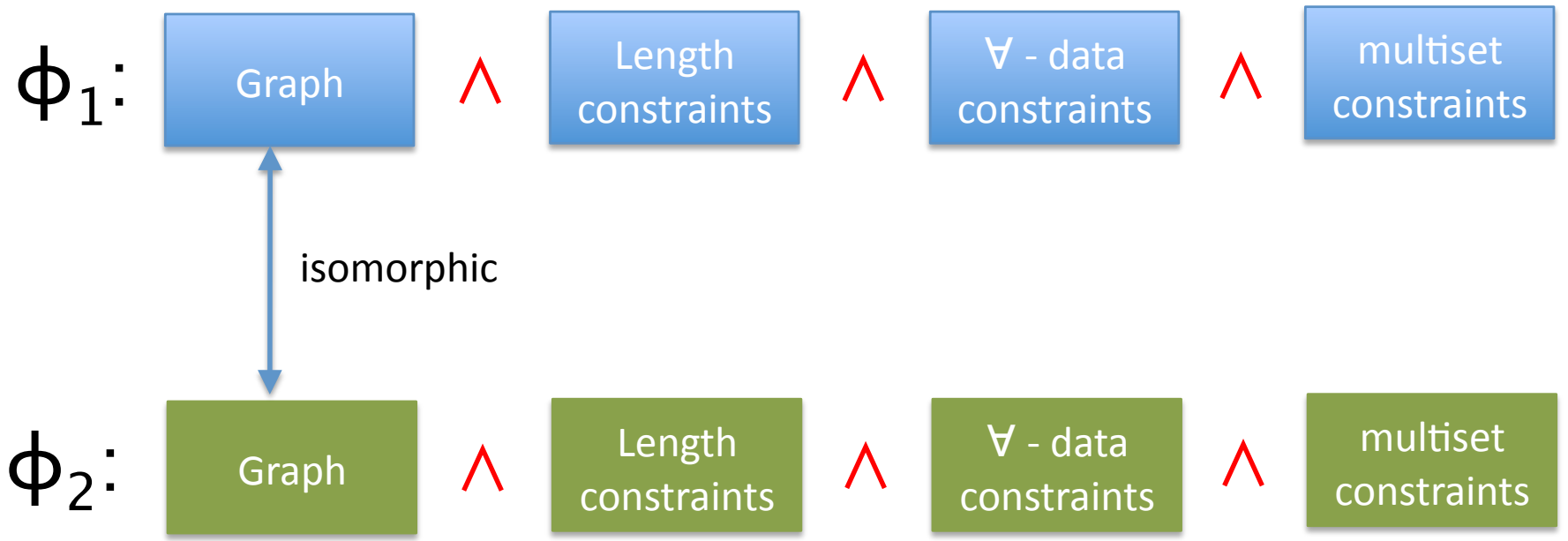


Join operator (case 1)

$\phi_1 \sqcup \phi_2 =$

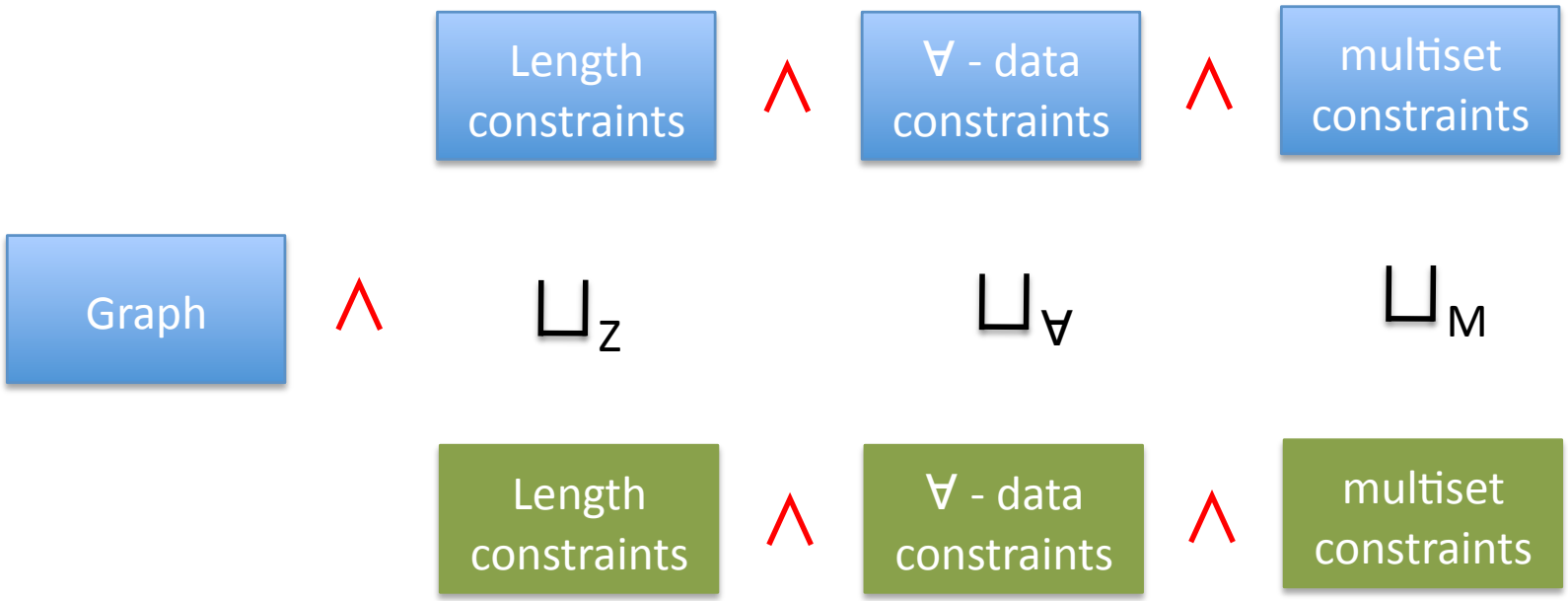


Join operator (case 2)



Join operator (case 2)

$\phi_1 \sqcup \phi_2 =$



Join operator on ∀-formulas

$\forall\ y1, y2 \in (x,\#).\ y1 \leq y2 \Rightarrow (x,n)[y1] > (x,n)[y2]$

$\forall\ y1 \in (x,\#)\ .\ (x,n)[y1] >= 6$

$\sqcup_{\forall}$

$\forall\ y1, y2 \in (x,\#).\ y1 \leq y2 \Rightarrow (x,n)[y1] >= (x,n)[y2]$

$\forall\ y1 \in (x,\#)\ .\ (x,n)[y1] >= 2$

$=$

$\forall\ y1, y2 \in (x,\#).\ y1 \leq y2 \Rightarrow (x,n)[y1] >= (x,n)[y2]$

$\forall\ y1 \in (x,\#)\ .\ (x,n)[y1] >= 2$

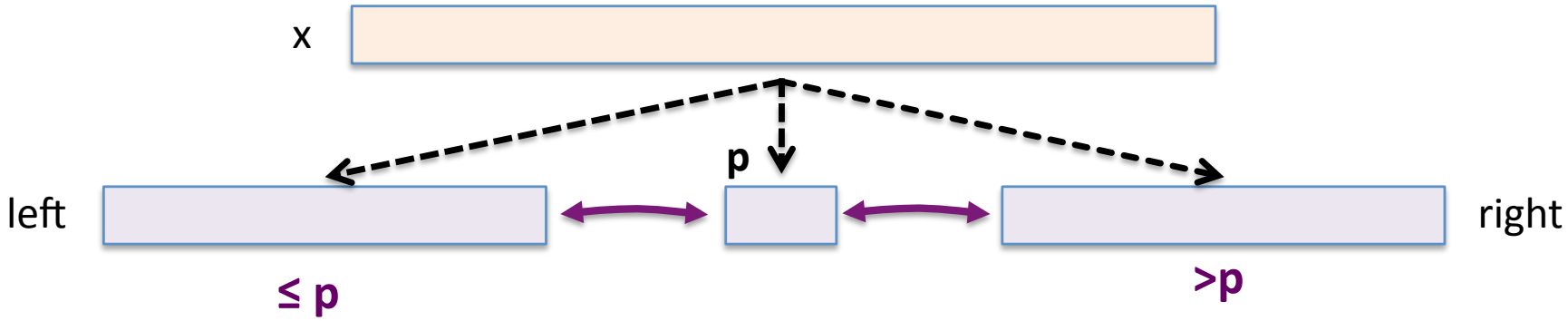
Analysis

- Basic mechanism: Unfold/Fold
  - Unfold: Post-images introduce new nodes
  - Fold: Eliminate nodes and collect information

  ➔ *Generation of universally quantified formulas*

- Procedures: Modular analysis
  - Compute local heap transformations
  - Call/Return: Eliminate old heap?

Iterative split: analysis with $\mathfrak{A}_{HS}(\mathfrak{A}_U)$



"all values are less than the pivot"

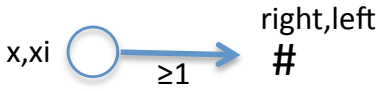
$\forall y \in \text{left}.\ \text{left}[y] \leq \text{head}(p)$

Parameters of $\mathfrak{A}_U$:

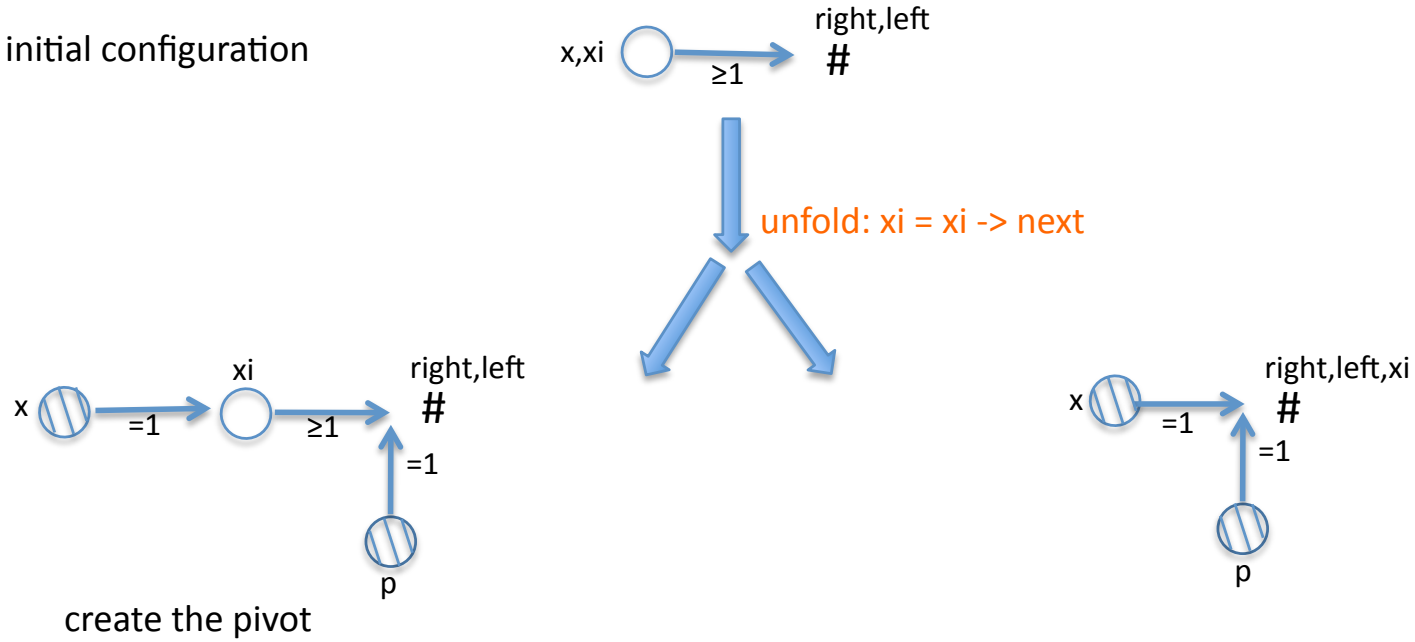
- polyhedra domain
- patterns: $\forall\, y \in \bigcirc$

Iterative split: analysis with $\mathfrak{A}_{HS}(\mathfrak{A}_U)$

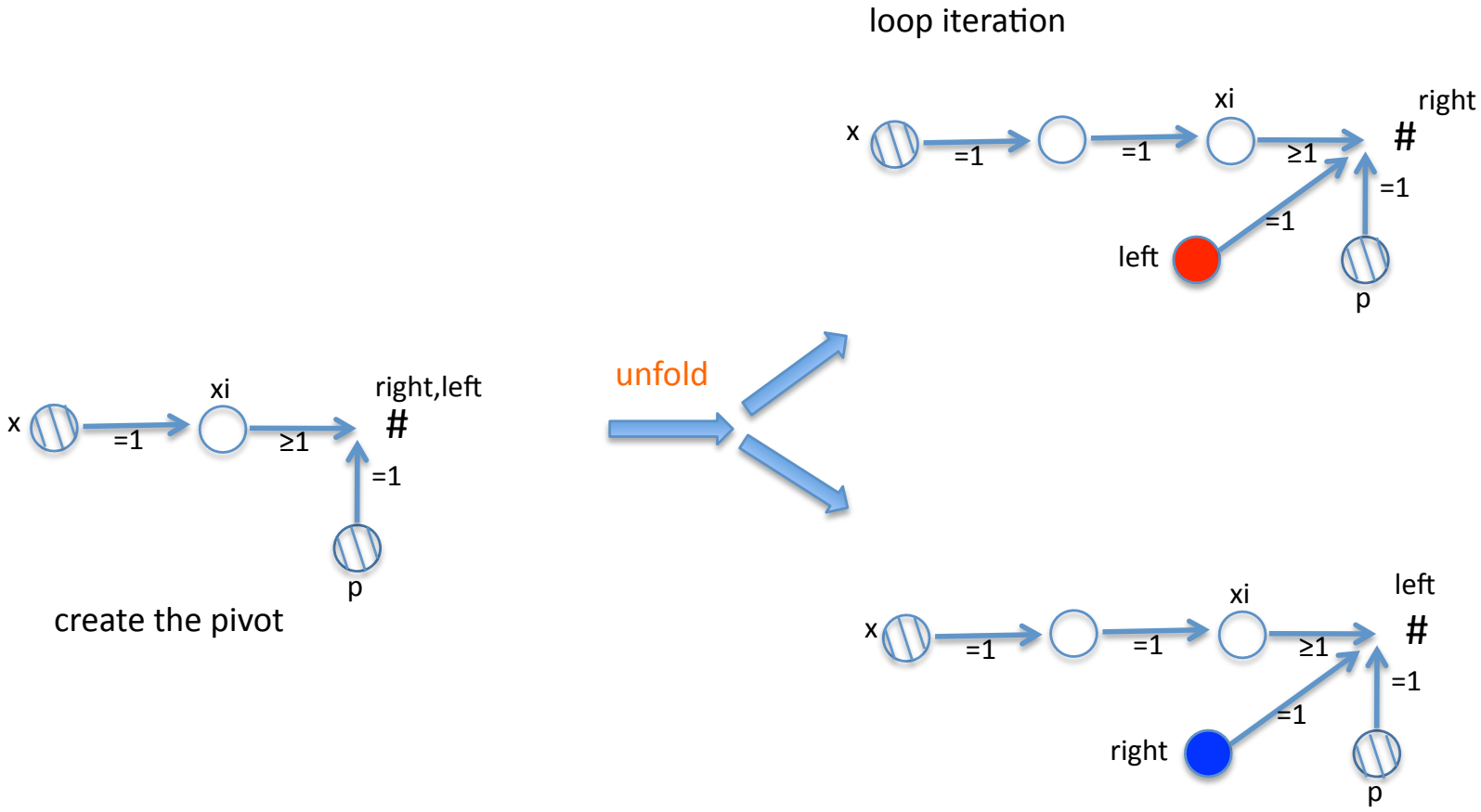
initial configuration



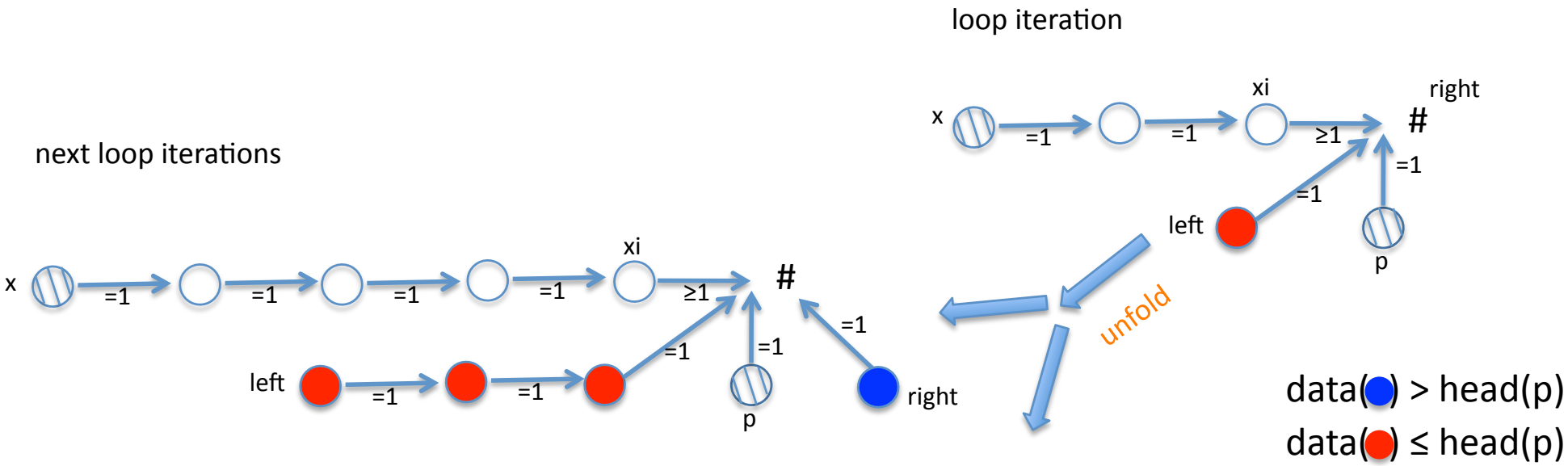
Iterative split: analysis with $\mathfrak{A}_{HS}(\mathfrak{A}_U)$



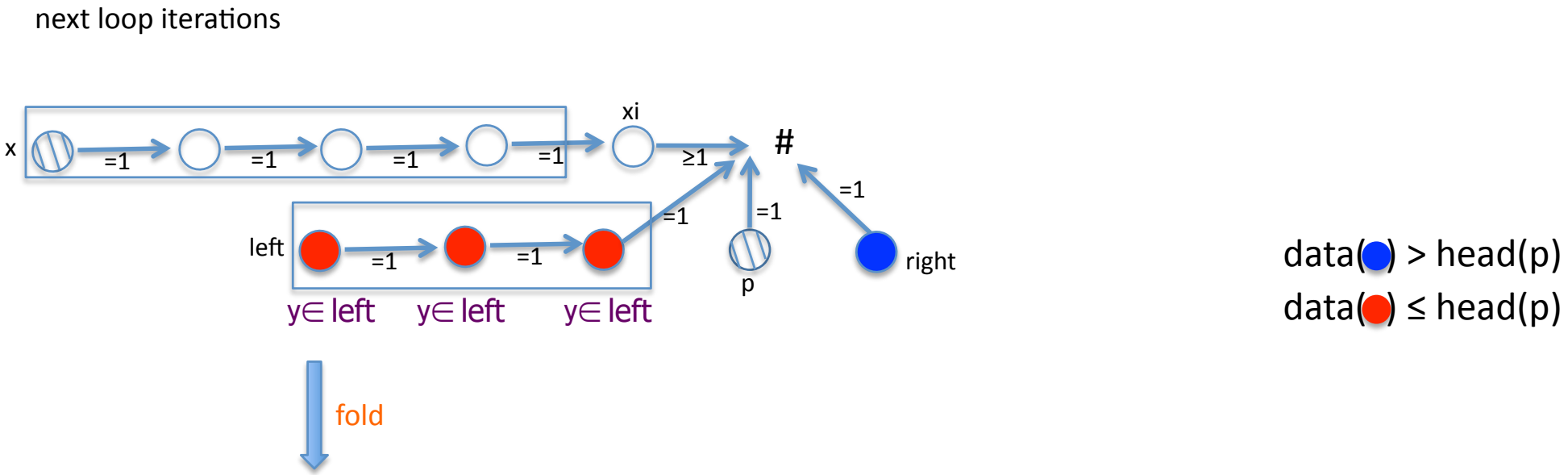
Iterative split: analysis with $\mathfrak{A}_{HS}(\mathfrak{A}_U)$



Iterative split: analysis with $\mathfrak{A}_{HS}(\mathfrak{A}_U)$

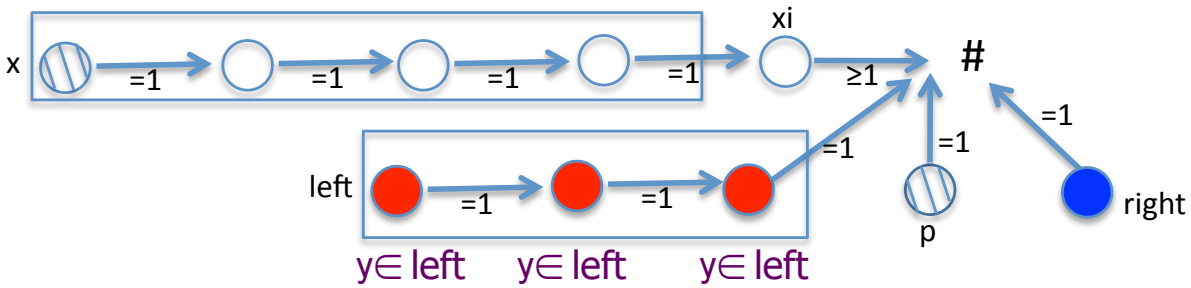


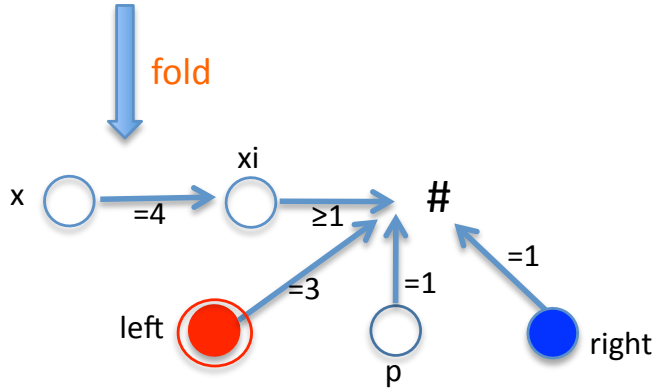
Iterative split: analysis with $\mathfrak{A}_{HS}(\mathfrak{A}_{U})$



Iterative split: analysis with $\mathfrak{A}_{HS}(\mathfrak{A}_U)$

next loop iterations



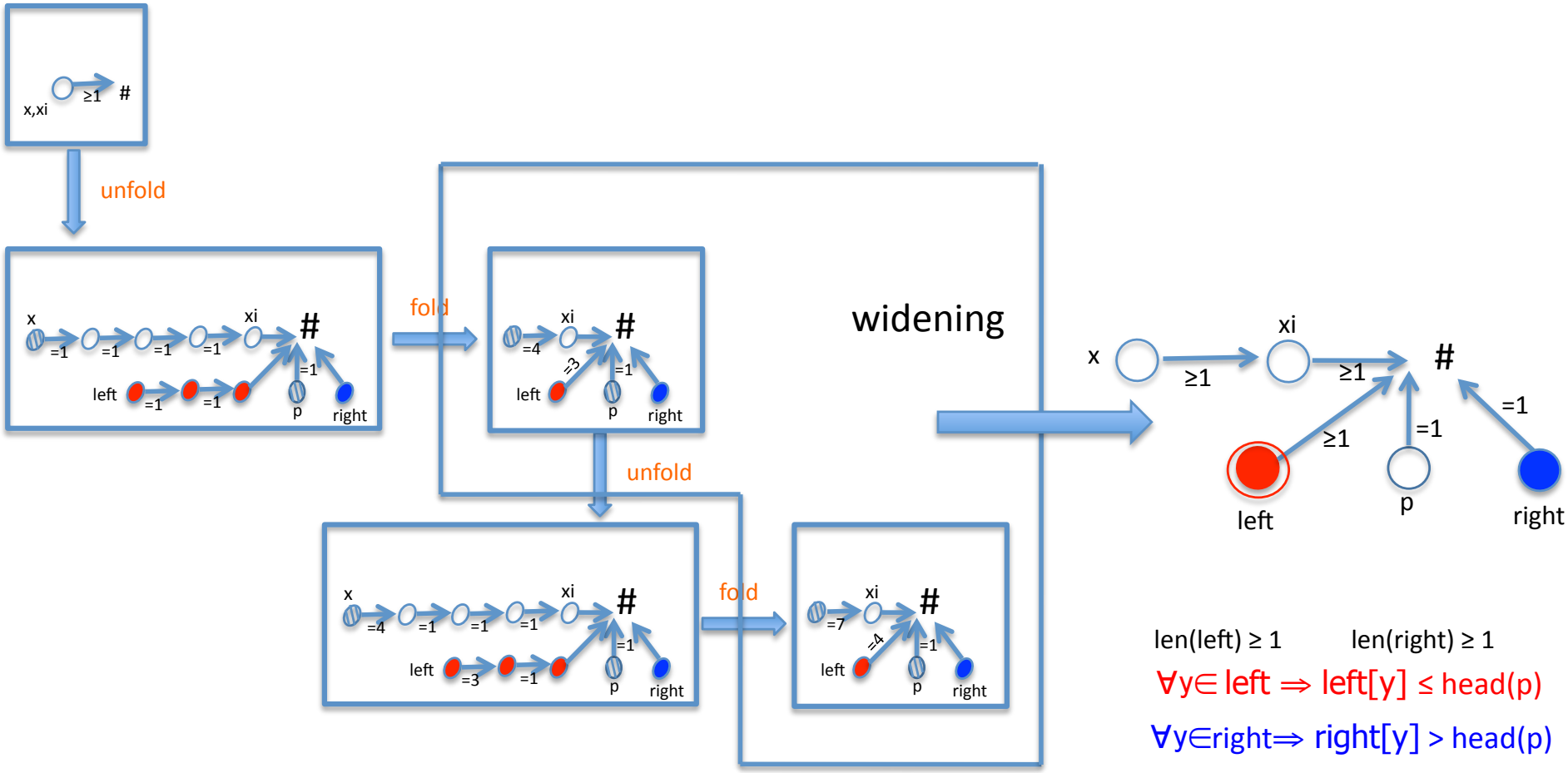


data( ) > head(p)

data( ) ≤ head(p)

$\forall y \in left \Rightarrow left[y] \leq head(p)$

Iterative split: analysis with $\mathfrak{A}_{HS}(\mathfrak{A}_U)$



N -ary Patterns?

Concatenation ➔ loss of information

For all y1, y2. (y2 = y1+1) => segm[y2] – segm[y1] = 2

- A prefix has the property
- Unfold: 2 more elements have the property
- How to link with the previous prefix?
- Need information on boundary elements

➔ Include more Patterns: Closure

Outline

- Logic SL3
- SL3: Entailment checking
- Abstract Domains for Heaps with Data
- Experimental results (CELIA)

Celia

www.liafa.jussieu.fr/celia

- Plug-in of FRAMA-C platform for the analysis of C programs.
  http://frama-c.com/

- FIXPOINT : a generic fixpoint engine [B. Jeannet]

- APRON: numerical abstract domains library
  [B. Jeannet and A. Mine]

Benchmark:

- Basic functions in usual libraries on singly-linked lists
- Programs calling functions on singly-linked lists

Experimental results

class	*fun*	A_M t (s)	A_U *P*	A_U t (s)	*Examples of summaries synthesized*
sll	*create*	0.013	$G_=, G_1$	0.021	
	addfst	0.003	$G_=$	0.002	
	addlst	0.031	$G_=$	0.033	$\rho^{\#}_U(create(\&x,\ell))$: $\mathbf{hd}(x) = 0 \wedge \mathbf{len}(x) = \ell \wedge \forall y \in (x) \Rightarrow d[y] = 0$
	delfst	0.001	$G_=$	0.001	
	dellst	0.034	$G_=$	0.042	
map	*init*(v)	0.024	$G_=, G_1$	0.034	$\rho^{\#}_U(init(v,x))$: $\mathbf{len}(x^0) = \mathbf{len}(x) \wedge \mathbf{hd}(x) = v \wedge \forall y \in (x).\ d[y] = v$
	init_even(k)	0.024	G_1	0.034	$\rho^{\#}_U(init\_even(k))$: $\mathbf{len}(r) = k \wedge \mathbf{hd}(r) = 0 \wedge \forall y \in (r).\ d[y] = 2 * y$
	add$_k$(v)	0.021	$G_=$	0.032	
map2	*add*$_k$(v)	0.089	$G_=$	0.517	$\rho^{\#}_U(add_k(v,x,z))$: $\mathbf{len}(x^0) = \mathbf{len}(x) \wedge \mathbf{len}(z^0) = \mathbf{len}(z) \wedge eq_\forall(x,x^0) \wedge$
	copy	0.063	$G_=$	0.078	$\forall y_1 \in x, y_2 \in z.\ y_1 = y_2 \Rightarrow d[y_1] + v = d[y_2]$
fold	*del*Pred	0.062	$G_=, G_1$	0.145	$\rho^{\#}_M(split(v,x,\&l,\&u))$: $\mathbf{ms}(x) = \mathbf{ms}(x^0) = \mathbf{ms}(l) \cup \mathbf{ms}(u)$
	max	0.031	$G_=, G_1$	0.048	
	clone(x)	0.071	$G_=$	0.315	$\rho^{\#}_U(clone(x))$: $\mathbf{len}(x^0) = \mathbf{len}(x) = \mathbf{len}(r)$
	split	0.245	$G_=, G_1$	0.871	$\wedge\ \forall y_1 \in x, y_2 \in r.\ y_1 = y_2 \Rightarrow d[y_1] = d[y_2]$
fold2	*equal*	0.127	$G_=$	0.261	$\rho^{\#}_M(merge(x,z,\&r))$: $\mathbf{ms}(x) \cup \mathbf{ms}(z) = \mathbf{ms}(r) \wedge \mathbf{ms}(x^0) = \mathbf{ms}(x) \wedge \ldots$
	concat	0.217	$G_=, G_1, G_2$	0.806	$\rho^{\#}_U(merge(x,z,\&r))$: $eq_\forall(x,x^0) \wedge eq_\forall(z,z^0) \wedge sorted(x^0) \wedge sorted(z^0) \wedge$
	merge	1.014	$G_=, G_1, G_2$	2.306	$sorted(r) \wedge \mathbf{len}(x) + \mathbf{len}(z) = \mathbf{len}(r)$
sort	*bubble*	0.387	$G_=, G_1, G_2$	2.190	
	insert	0.557	$G_=, G_1, G_2$	3.292	$\rho^{\#}_M(quicksort(x))$: $\mathbf{ms}(x) = \mathbf{ms}(x^0) = \mathbf{ms}(res)$
	quick	1.541	G_1, G_2	121.1	$\rho^{\#}_U(quicksort(x))$: $eq_\forall(x,x^0) \wedge sorted(res)$
	merge	1.547	$G_=, G_1, G_2$	95.94	

Summary

- AI-based framework for proving assertions:
  - SL3 formulas ➔ Abstract objects
  - Abstract domains for heaps and data
- Reasoning about different kinds of constraints
- Key operations: Node elimination (Fold and Strengthen)
  - *Fold*: collecting information in a sequence
  - *Strengthen*: Information transfer
- Sound Entailment Checking/SAT for SL3
- Powerful and efficient techniques, tool CELIA

Ongoing/Future Work

- More general data structures
  - Doubly-linked lists, Trees
  - Composite data structures
- Sound and accurate decision procedures
  - Cover general classes of constraints
  - Simple abstract domains + composition
  - Completeness for well identified fragments
- Concurrent programs
  - Compositional reasoning
  - Sequentialization