String Solving Strategies

Straight-Line Fragment

Simplified Syntax

Rewrite constraints to use simple terms. Conjunctions of

- concatenation: $X = Y\ Z$
- regular tests: $X \in r$

Write a sequences ϕ, Ψ instead of conjunctions $\phi \wedge \Psi$

E.g.

$$b\ X = X\ c\ Y \wedge X \in ab^*c$$

becomes

$$Z = B\ X,\quad B \in b,\quad Z = X\ Z',\quad Z' = C\ Y,\quad C \in c,\quad X \in ab^*c$$

Boolean Combinations

We can generalise to arbitrary boolean combinations of terms

$$\phi, \phi' \;:=\; X = Y\,Z \mid X \in r \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid \neg \phi$$

Two quick ways to see this:

- Disjoint Normal Form
  - Rewrite ϕ into a disjunction of conjunctions
  - Consider each conjunction in turn
- DPLL[T] / SMT solving
  - Explore Boolean structure via DPLL algorithm
  - Test feasibility of conjunctive queries

Proof System

Our approach to solving constraints is to use a proof system

1. start with a constraint
2. apply proof rules
3. find a contradiction or a solution

This gives a search problem: apply rules until you find the answer

Rules applied non-deterministically

We will discuss deterministic strategies at the end

Proof Rules Syntax

A simple proof rule is

$$\frac{\ldots, \Phi, \Psi}{\ldots, \Phi \wedge \Psi}$$

(read upwards)

If we have $\Phi \wedge \Psi$ we can derive both Φ and Ψ

A rule that creates two branches:

- find a solution on one of them, or
- find a contradiction on all of them

$$\frac{\ldots, \Phi \qquad\qquad \ldots, \Psi}{\ldots, \Phi \vee \Psi}$$

Constraint Propagation

For each

$$X = Y\ Z$$

Forwards propagation

- Push constraints on Y and Z into a constraint on X

Backwards propagation

- Pull constraint on X back into constraints on Y and Z

String Functions

Concatenation is a string function

$$X = Y\ Z \longrightarrow X = \mathrm{concat}(Y, Z)$$

More generally, any function could appear

$$X = f(X_1, \ldots, X_n)$$

where f is some function with input $X_1, \ldots, X_n$.

Forwards Propagation

Forwards Propagation (concat)

Suppose

$$\ldots,\ X = Y\,Z,\ \ Y \in r_1,\ \ Z \in r_2,\ \ldots$$

We can infer that also

$$X \in r_1\, r_2$$

Or in fact

$$X \in r$$

L(r) -- set of words matching r

for any r with

$$L(r) = \mathrm{concat}(L(r_1), L(r_2))$$

As a Proof Rule

(new bit)

$$\frac{\ldots,\quad X \in r,\quad X = Y\,Z,\quad Y \in r_1,\quad Z \in r_2,\ \ldots}{\ldots,\ X = Y\,Z,\quad Y \in r_1,\ \ Z \in r_2,\ \ldots} \qquad L(r) = L(r_1\ r_2)$$

Forwards Propagation in General

Suppose

$$\ldots, \quad X = f(X_1, \ldots, X_n), \quad X_1 \in r_1, \ldots, \quad X_n \in r_n, \ldots$$

and

$$f(L(r_1), \ldots, L(r_n)) = L(r) \text{ for some regular expression } r$$

Add constraint $X \in r$

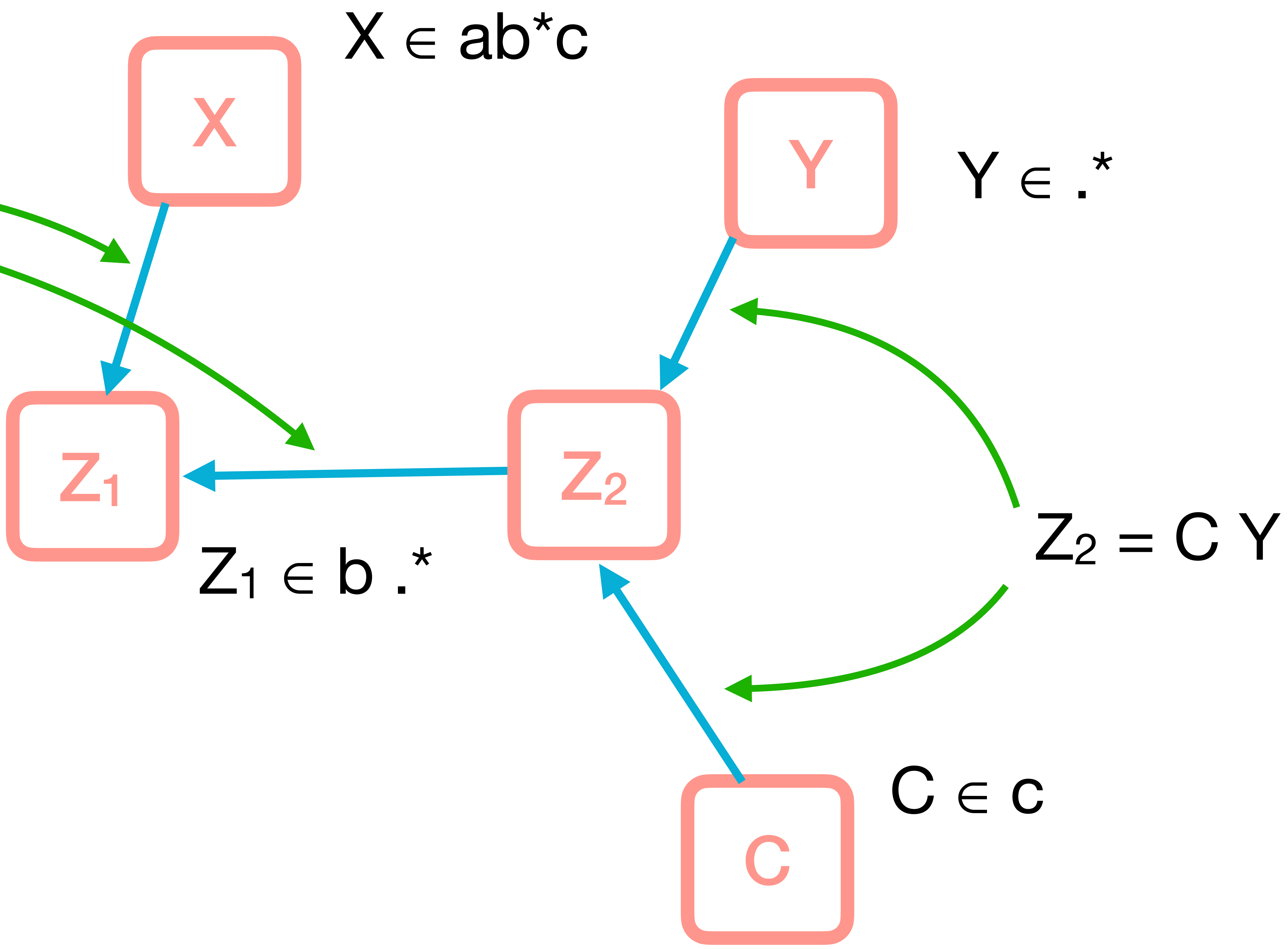
$$\ldots, \underline{X \in r,} \quad X = f(X_1, \ldots, X_n), \quad X_1 \in r_1, \ldots, \quad X_n \in r_n, \ldots$$

As a Proof Rule

(new bit)

$$\frac{\ldots,\quad X \in r,\quad X = f(X_1, \ldots, X_n),\quad X_1 \in r_1,\quad \ldots,\quad X_n \in r_n,\ \ldots}{\ldots,\quad X = f(X_1, \ldots, X_n),\quad X_1 \in r_1, \ldots,\quad X_n \in r_n,\ \ldots}\quad L(r) = f(L(r_1), \ldots, L(r_n))$$

Visual Example



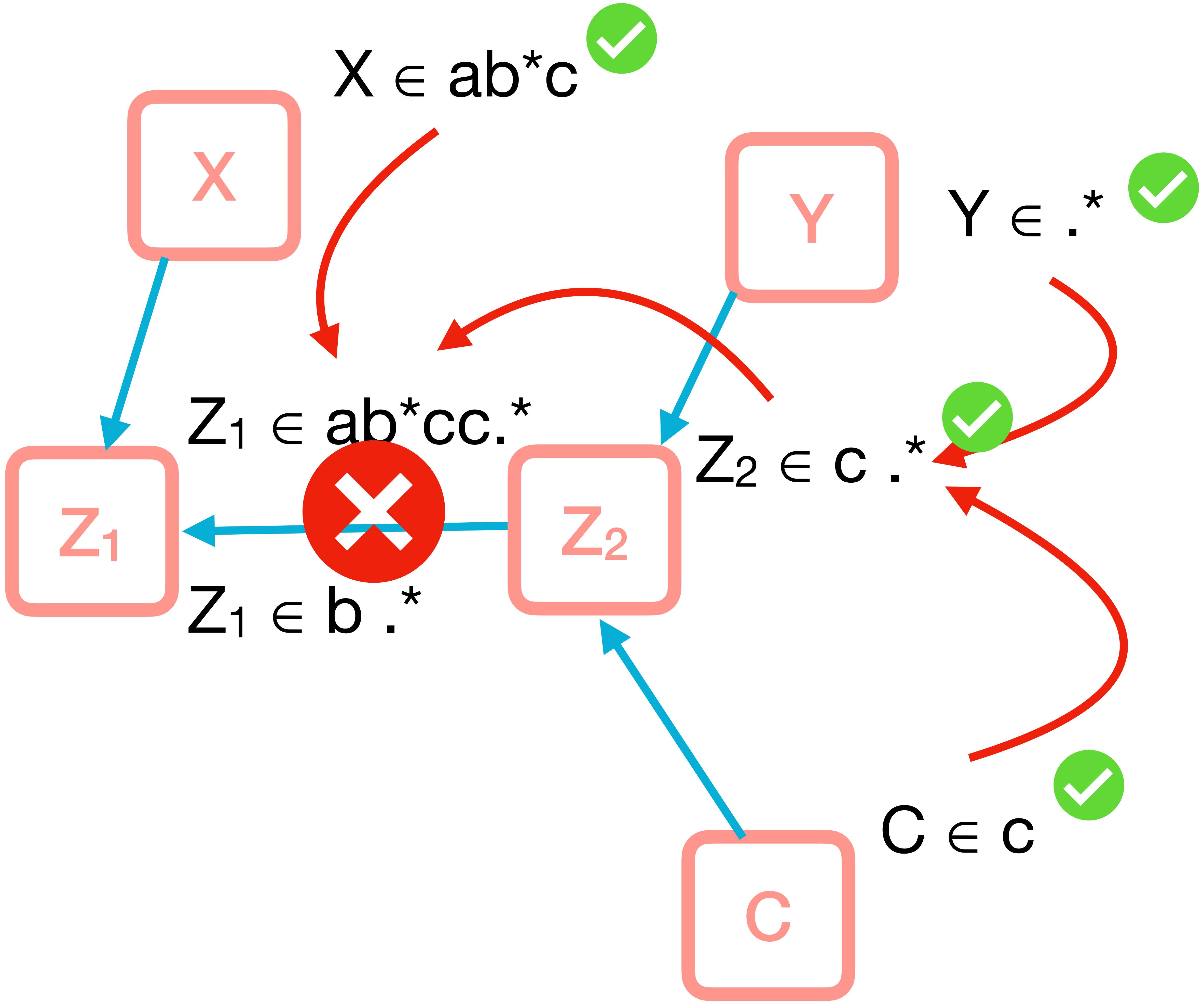
$\boxed{Z_1 = X\ Z_2}\ \boxed{Z_2 = C\ Y,}$

$C \in c, \quad X \in ab^*c, \quad Z_1 \in b\ .^*$

Solve via Forwards Propagation

$\boxed{Z_1 = X\, Z_2,}\ \boxed{Z_1 = C\, Y,}$

$C \in c, \quad X \in ab^*c, \quad Z_1 \in b\,.^*$



Now check satisfiability for each variable

Constraint Elimination?

Suppose

$$X = Y\,Y, \quad Y \in a^*b^*$$

Notice

$$L(a^*b^*a^*b^*) = concat(L(a^*b^*), L(a^*b^*))$$

Propagate

$$X \in a^*b^*a^*b^*, \;\; X = Y\,Y, \;\; Y \in a^*b^*$$

Do we still need X = Y Y?

An Overapproximation

$$X \in a^*b^*a^*b^*, \quad X = Y\,Y, \quad Y \in a^*b^*$$

Notice

- X = abaabb satisfies $X \in a^*b^*a^*b^*$
- But not $X = Y\,Y$ for any value of Y

We need to keep $X = Y\,Y$ too

Backwards Propagation

Backwards Propagation (concat)

Suppose

$$\ldots,\quad X \in r_1\, r_2,\quad X = Y\, Z,\quad \ldots$$

We can infer that also

$$Y \in r_1,\quad Z \in r_2$$

Or in fact

$$Y \in r_3,\quad Z \in r_4$$

for any r_3, r_4 with

$$\mathrm{concat}(L(r_3), L(r_4)) = L(r_1\, r_2)$$

Backwards Propagation

Take constraint

$$X \in ac^*d^* \wedge X = Y\,Z$$

Pull the constraint on X back through $X = Y\,Z$

$Y \in \varepsilon, \quad Z \in ac^*d^*$

$Y \in a, \quad Z \in c^*d^*$

$Y \in ac^*, \quad Z \in c^*d^*$

$Y\,Z \in ac^*d^*$

$X \in ac^*d^*$

$Y \in ac^*, \quad Z \in d^*$

$Y \in ac^*d^*, \quad Z \in \varepsilon$

Other String Functions

Backwards propagation creates branching for concat

What other string functions can be supported?

$X \in \varepsilon, \quad Y \in ac^*d^*$

$X \in a, \quad Y \in c^*d^*$

$X \in ac^*, \quad Y \in c^*d^*$ $\qquad XY \in ac^*d^*$

$X \in ac^*, \quad Y \in d^*$

$X \in ac^*d^*, \quad Y \in \varepsilon$

Conditions on String Functions

Consider

$$Z = \mathrm{concat}(X, Y)$$

Given a regular language L, what is $\mathrm{concat}^{-1}(L)$?

$X \in \varepsilon, \quad Y \in ac^*d^*$

$X \in a, \quad Y \in c^*d^*$

$X \in ac^*, \quad Y \in c^*d^*$

$X \in ac^*, \quad Y \in d^*$

$X \in ac^*d^*, \quad Y \in \varepsilon$

$XY \in ac^*d^*$

A finite union of regular alternatives

$$\bigcup_i L^i_X \times L^i_Y$$

(A recognisable relation)

Conditions on String Functions

Consider

$$X = f(X_1, \ldots, X_n)$$

Given a regular language L, require $f^{-1}(L)$ is recognisable

$X_1 \in r^1_1, \quad \ldots, \quad X_n \in r^1_n$

$X_1 \in r^2_1, \quad \ldots, \quad X_n \in r^2_n$

$X_1 \in r^3_1, \quad \ldots, \quad X_n \in r^3_n$

$X_1 \in r^4_1, \quad \ldots, \quad X_n \in r^4_n$

$X_1 \in r^5_1, \quad \ldots, \quad X_n \in r^5_n$

$f(X_1, \ldots, X_n) \in r$

$$f^{-1}(L) = \bigcup_i L^i_1 \times \ldots \times L^i_n$$

Proof Rule

Branching

$$\frac{\ldots, X \in r, X = Y\,Z, Y \in r^{1}{}_{2}, Z \in r^{1}{}_{2}, \ldots \qquad \ldots \qquad \ldots, X \in r, X = Y\,Z, Y \in r^{n}{}_{2}, Z \in r^{n}{}_{2}, \ldots}{\ldots, X \in r, X = Y\,Z, \ldots}$$

$L(r) = L(r^{i}{}_{1}\ r^{i}{}_{2})$

New bits

Example

$\phi := W \in b, \quad Y \in b^*c^*, \quad Z \in a^*bc^*,$

$X = W, \quad Z = X\,Y$

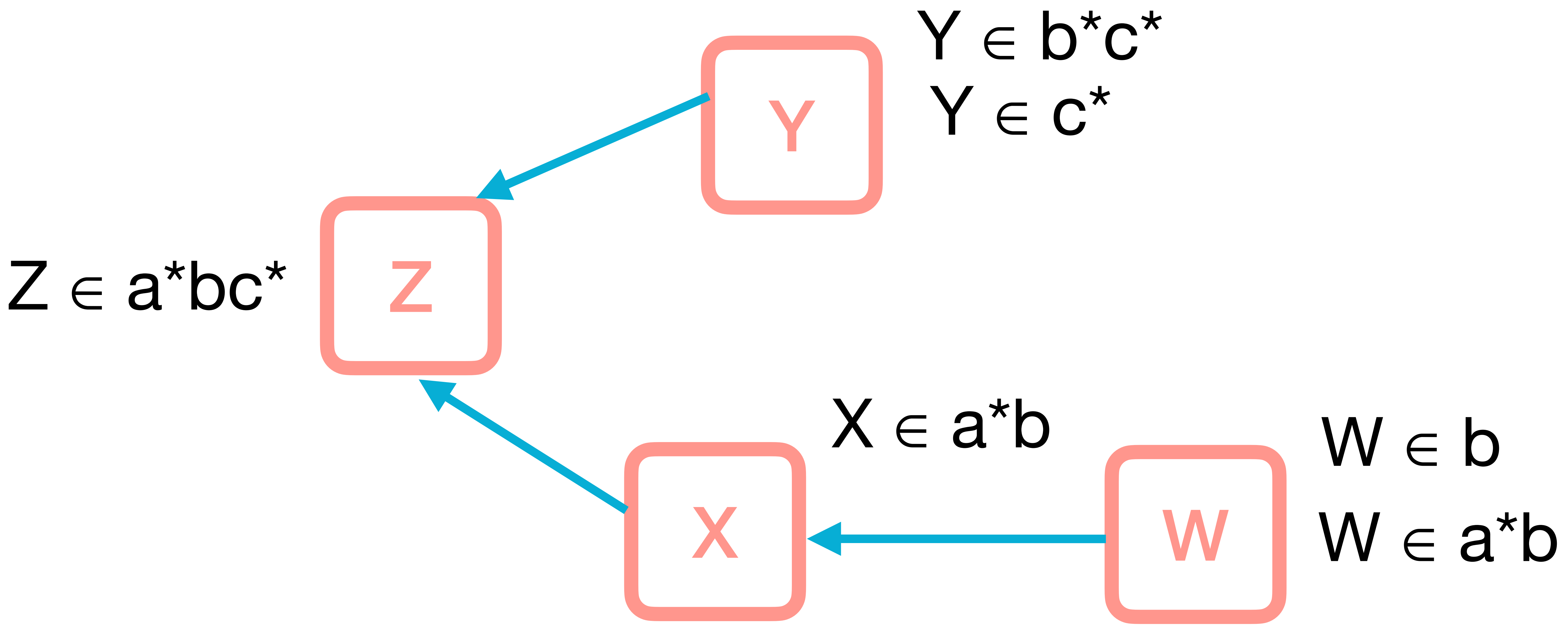
ϕ

$\downarrow$

$\phi,$
$X \in a^*b,$
$Y \in c^*$

$\downarrow$

$\phi,$
$X \in a^*b,$
$Y \in c^*,$
$W \in a^*b$



?

We have finished propagating

There are no contradictions -- are we done?

With Proof Rules

$$\dfrac{\dfrac{W \in b,\ Y \in b^*c^*,\ Z \in a^*bc^*,\ X = W,\ Z = X\ Y,\ X \in a^*b,\ Y \in c,\ X \in a^*b,\ Y \in c^*,\ W \in a^*b}{W \in b,\ Y \in b^*c^*,\ Z \in a^*bc^*,\ X = W,\ Z = X\ Y,\ X \in a^*b,\ Y \in c} \qquad \ldots}{W \in b,\ Y \in b^*c^*,\ Z \in a^*bc^*,\ X = W,\ Z = X\ Y}$$

Other branches

Extracting Solutions

Extracting Solutions

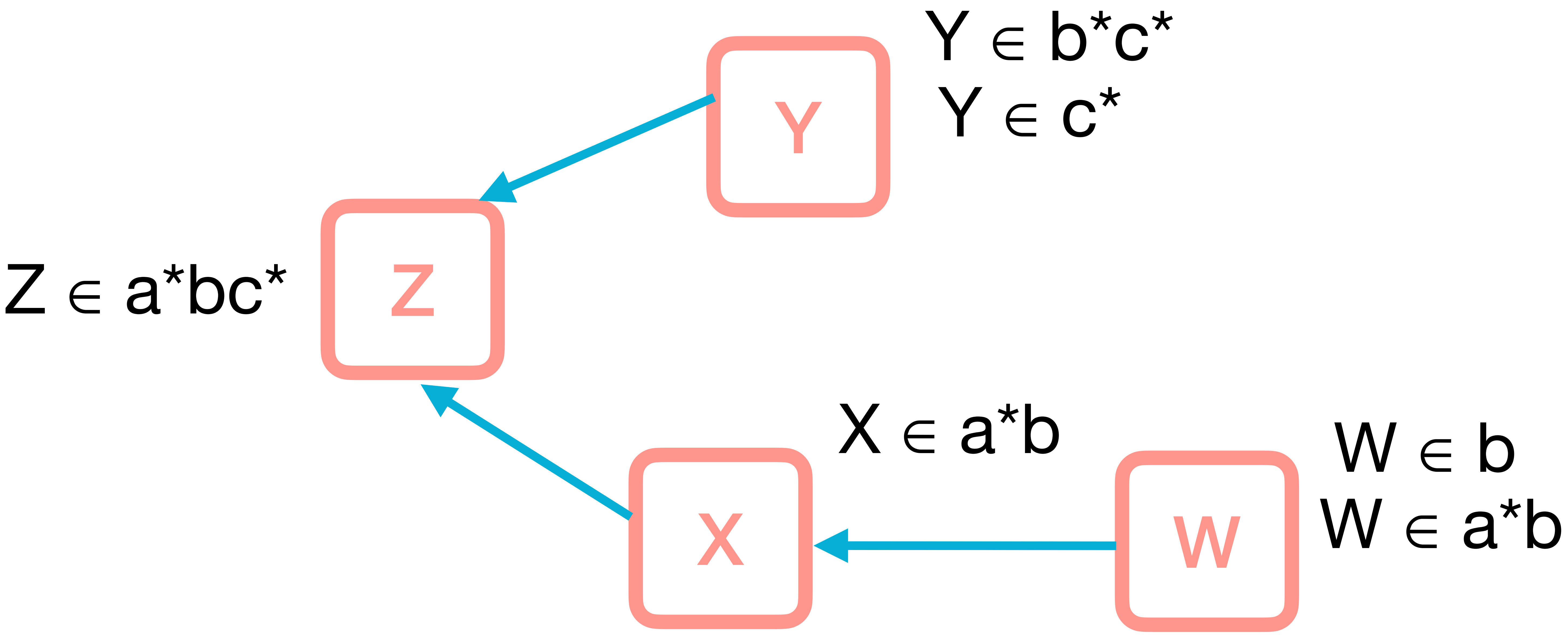
$W \in b, \quad Y \in b^*c^*, \quad Z \in a^*bc^*,$

$X = W, \quad Z = X\,Y$

Not all solutions to the regular constraints are correct

E.g.

$X = ab$
$Y = c$
$Z = b$
$W = b$



Satisfies each node but not constraint

$(Z \,!\!= X\,Y)$

A Cut Rule

As part of the proof search we can introduce two branches

$$X \in r \qquad\qquad X \notin r$$

for some regular expression r

Can be used to guess a solution

- Pick some w consistent with constraints on X
- Add $X \in w$ on one branch, $X \notin w$ on the other

Then propagate to other variables

As a Proof Rule

$$\frac{\ldots, X \in r \qquad\qquad \ldots, X \notin r}{\ldots}$$

Closing Branches

Constraints introduced via cut may be consistent or not

Close a branch if an inconsistency is found

$$\ldots,\quad X \in r_1,\quad \ldots,\quad X \in r_n,\quad \ldots \quad \text{with} \quad L(r_1) \cap \ldots \cap L(r_n) = \varnothing$$

Solution found if we have a "pure" solution

$$X_1 \in w_1,\quad \ldots,\quad X_n \in w_n$$

But how do we eliminate constraints?

As Proof Rules

Contradiction:

$$\frac{}{\ldots,\ \ X \in r_1,\ \ \ldots,\ \ X \in r_n}\ L(r_1) \cap \ldots \cap L(r_n) = \varnothing$$

Solution:

$$\frac{}{X_1 \in w_1,\ \ \ldots,\ \ X_n \in w_n}$$

Eliminating Constraints

To end up with a pure solution

$$X_1 \in w_1, \quad \dots, \quad X_n \in w_n$$

We need to eliminate all other constraints.

Two methods

- Eliminate multiple regular constraints: $X \in r_1, \quad \dots, \quad X \in r_n$
- Eliminate assignments: $X = f(X_1, \dots, X_n)$

Eliminate Multiple Constraints

$$X \in r_1, \quad \ldots, \quad X \in r_n$$

1. Replace with its intersection

$$X \in r_1 \cap \ldots \cap r_n$$

2. Remove a subsumed constraint

$$X \in r_2, \quad \ldots, \quad X \in r_n \qquad \text{when} \qquad L(r_2) \cap \ldots \cap L(r_n) \subseteq L(r_1)$$

Proof Rules

Intersection:

$$\frac{\ldots,\ X \in r_1 \cap \ldots \cap r_n}{\ldots,\ X \in r_1, \quad \ldots, \quad X \in r_n}$$

Subsumed:

$$\frac{\ldots,\ X \in r_2, \quad \ldots, \quad X \in r_n}{\ldots,\ X \in r_1, \quad \ldots, \quad X \in r_n} \quad L(r_2) \cap \ldots \cap L(r_n) \subseteq L(r_1)$$

Eliminate Function Assignments

$$X = f(X_1, \ldots, X_n)$$

Eliminating version of forwards propagation

Suppose

$$\ldots,\quad X = f(X_1, \ldots, X_n),\quad X_1 \in r_1,\quad \ldots,\quad X_n \in r_n,\ \ldots$$

and

$$f(L(r_1), \ldots, L(r_n)) = L(r) \text{ for some regular expression } r$$

If $L(r)$ only has one solution, replace $X = f(X_1, \ldots, X_n)$ with $X \in r$

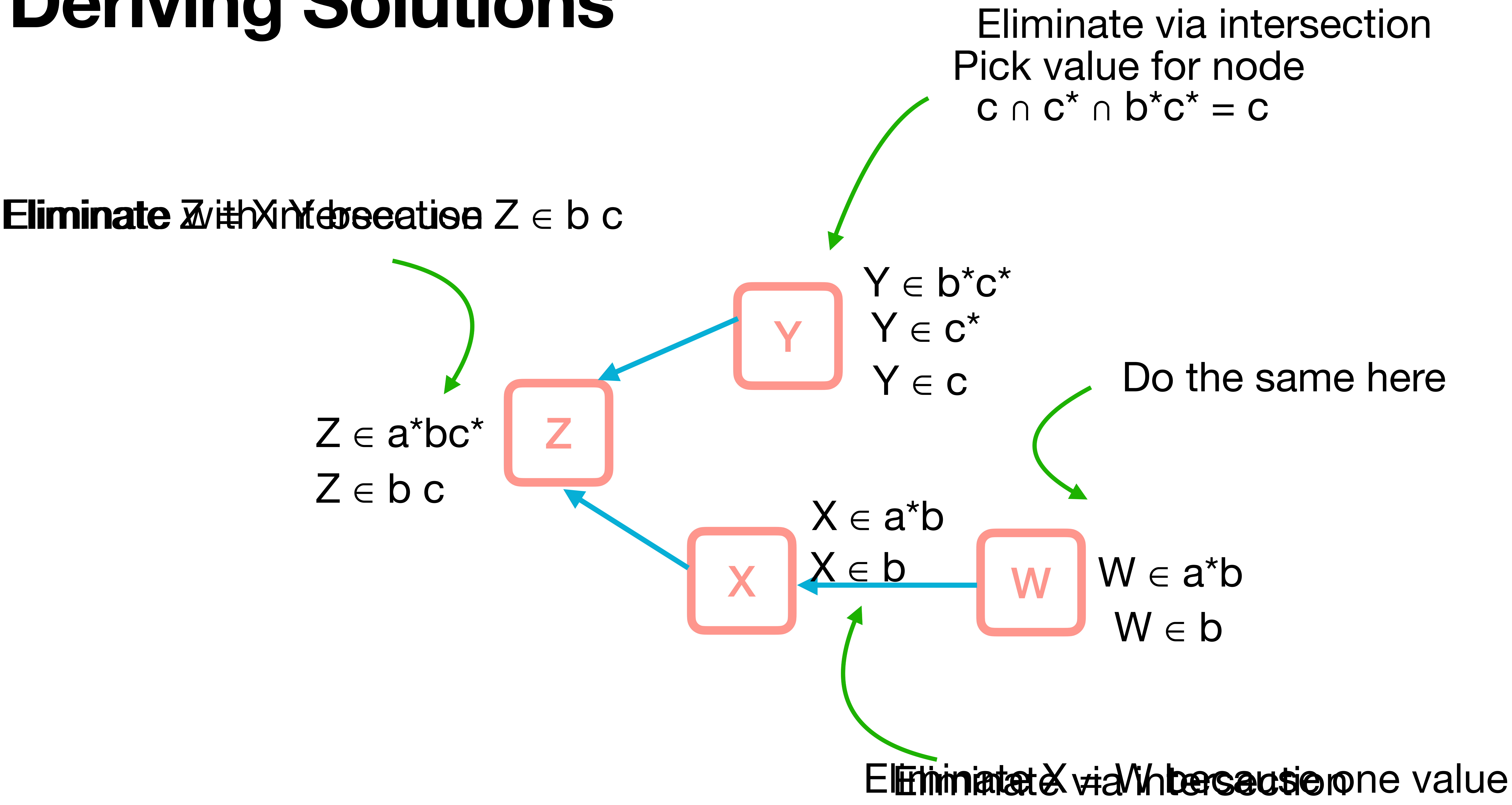
$$\ldots,\quad X \in r,\quad X_1 \in r_1,\quad \ldots,\quad X_n \in r_n,\quad \ldots$$

Proof Rules

(we dropped $X = f(X_1, \ldots, X_n)$)

$$\frac{\ldots,\ X \in r,\quad X_1 \in r_1,\quad \ldots,\quad X_n \in r_n,\ \ldots}{\ldots,\ X = f(X_1, \ldots, X_n),\quad X_1 \in r_1,\quad \ldots,\quad X_n \in r_n,\ \ldots} \quad \begin{array}{c} L(r) = f(L(r_1), \ldots, L(r_n)) \\ |L(r)| = 1 \end{array}$$

Deriving Solutions



With Proof Rules

$W \in b,\ X \in b,\ Z \in bc,\ Y \in c$

$\cap Z$

$W \in b,\ Z \in a^*bc^*,\ X \in b,\ Z \in bc,\ Y \in c,\ X \in a^*b$

F-Elim $Z = X\ Y$

$W \in b,\ Z \in a^*bc^*,\ X \in b,\ Z = X\ Y,\ Y \in c,\ X \in a^*b$

$\cap X$

$W \in b,\ Z \in a^*bc^*,\ X \in b,\ Z = X\ Y,\ X \in a^*b,\ Y \in c,\ X \in a^*b$

F-Elim $X=W$

$W \in b,\ Z \in a^*bc^*,\ X = W,\ Z = X\ Y,\ X \in a^*b,\ Y \in c,\ X \in a^*b$

$\cap W$

$W \in b,\ Z \in a^*bc^*,\ X = W,\ Z = X\ Y,\ X \in a^*b,\ Y \in c,\ X \in a^*b,\ W \in a^*b$

$\cap Y$

$W \in b,\ Y \in b^*c^*,\ Z \in a^*bc^*,\ X = W,\ Z = X\ Y,\ X \in a^*b,\ Y \in c,\ X \in a^*b,\ Y \in c^*,\ W \in a^*b$

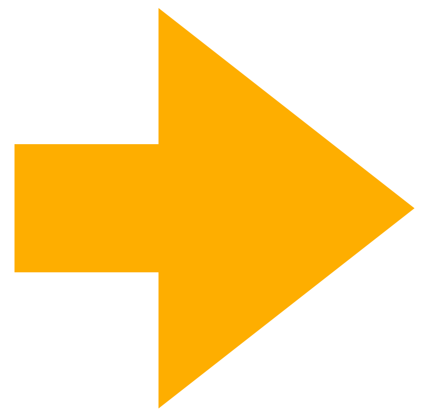
Forwards or Backwards?

Comparing Propagation Direction

We've seen forwards and backwards propagation.

Forwards:

- Regular inputs lead to regular outputs
- But had to overapproximate e.g. concatenation
- No branching required



Backwards:

- Regular outputs lead exactly to recognisable inputs
- No approximation
- Branching required

Functions Supporting Backwards

Some common string functions support backwards propagation

- Concatenation
- Reverse
- Replace(All)
  - X = ReplaceAll(Y, e_1, e_2)
  - e_1 -- a string or regular expression (with capture groups)
  - e_2 -- a string, or variable, or replacement pattern (using capture groups)
- Transductions (e.g. HTML escape)

Completeness

Proof Search

Our proof rules are sound

- We find only correct answers

Not necessarily complete

- Propagation may not terminate
- Cut rule introduces infinite alternatives

Currently we've shown proof rules but no strategy

That is, apply the rules non-deterministically to solve a constraint.

Can we provide an algorithm with guarantees?

Complete Fragments

Implementations needs to apply rules algorithmically

1. Apply forwards propagation until fixed point
   - Complete for constraints with the "tree property" [Kan etal, CPP 2022]
2. Apply backwards propagation once (then extract solutions)
   - Complete for "straight-line" (see later) [Chen et al, POPL 2022]
3. "Chain-free" fragment
   - More complex mix of forward and backwards [Abdulla etal, ATVA 2019]

Straight-Line Constraints

Straight Line Conjunctions

A constraint is straight-line if all of its conjunctions are straight-line

Conjunctions are "SSA" (Single Static Assignment): each variable assigned once.

A conjunction is straight-line if

- Each variable appears on the LHS at most once
- Variables can be ordered $X_1, \ldots, X_n$
    - If $X_i = \ldots X_j \ldots$ then $j < i$

Examples

Straight-line:

$X \in a^*b^*,$
$Y = X\,X,$
$Y \in abab^*,$
$Z = X\,Y,$
$Z \in ababab$

Not straight-line:

$X \in a^*b^*,$
$Y = X\,X,$
$Y \in abab^*,$
$Y = X\,Y$

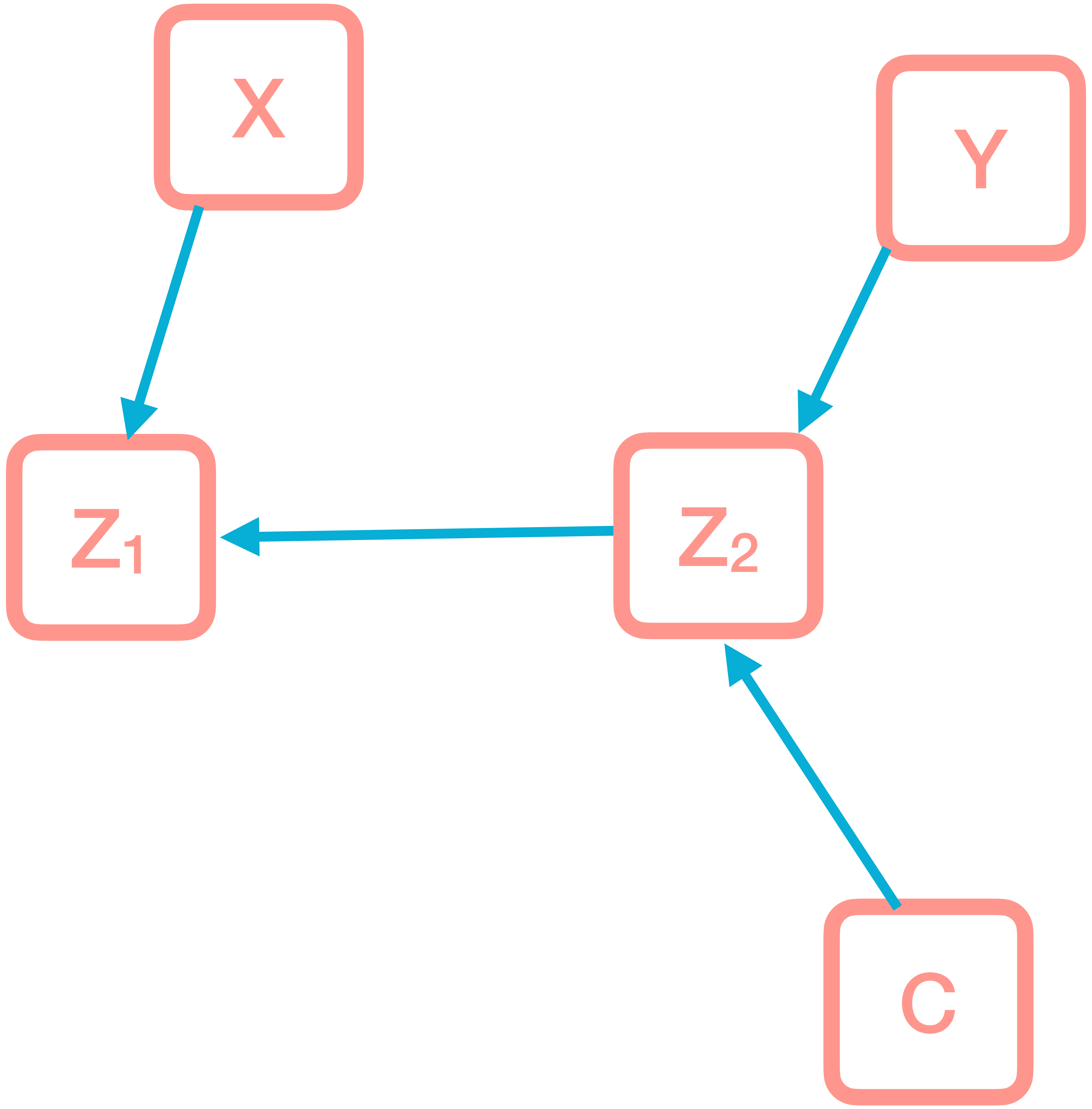
Not straight-line:

$X \in a^*b^*,$
$Y = X\,X,$
$Y \in abab^*,$
$X = Y\,Y$

Visual Example

$$Z_1 = X\,Z_2,\quad Z_2 = C\,Y,$$

$$C \in c,\quad X \in ab^*c,\quad Z_1 \in b\,.^*$$



Straight-line:

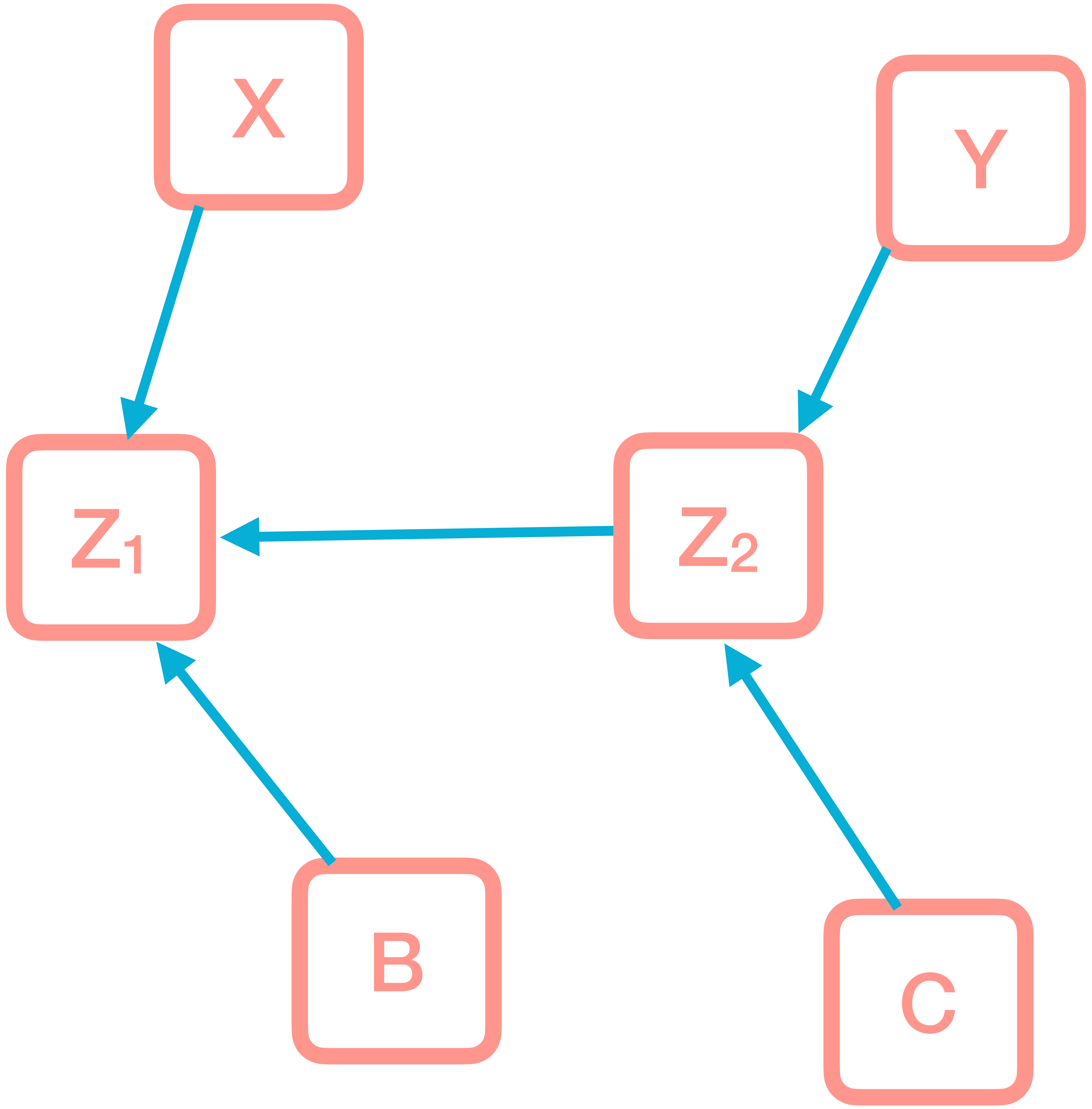
- No cycles
- Each node has incoming arrows from one constraint

Visual Non-Example 1

$$Z_1 = X\,Z_2, \quad Z_2 = C\,Y,$$

$$C \in c, \quad X \in ab^*c, \quad Z_1 \in b\,.^*,$$

$$Z_1 = Z_2\,B$$



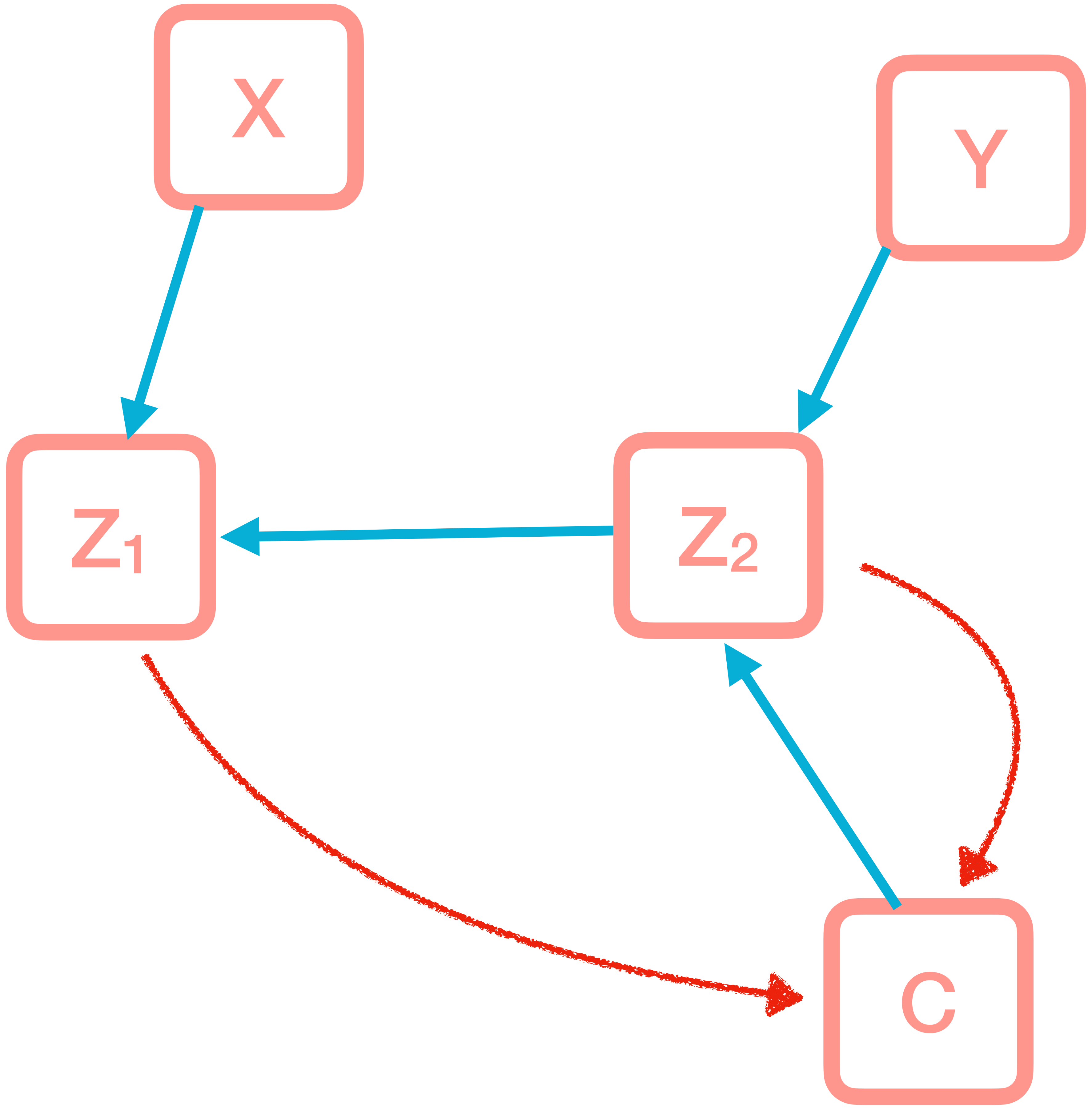
Z1 has two incoming constraints

Visual Non-Example 2

$Z_1 = X\,Z_2, \quad Z_2 = C\,Y,$

$C \in c, \quad X \in ab^*c, \quad Z_1 \in b\ .^*,$

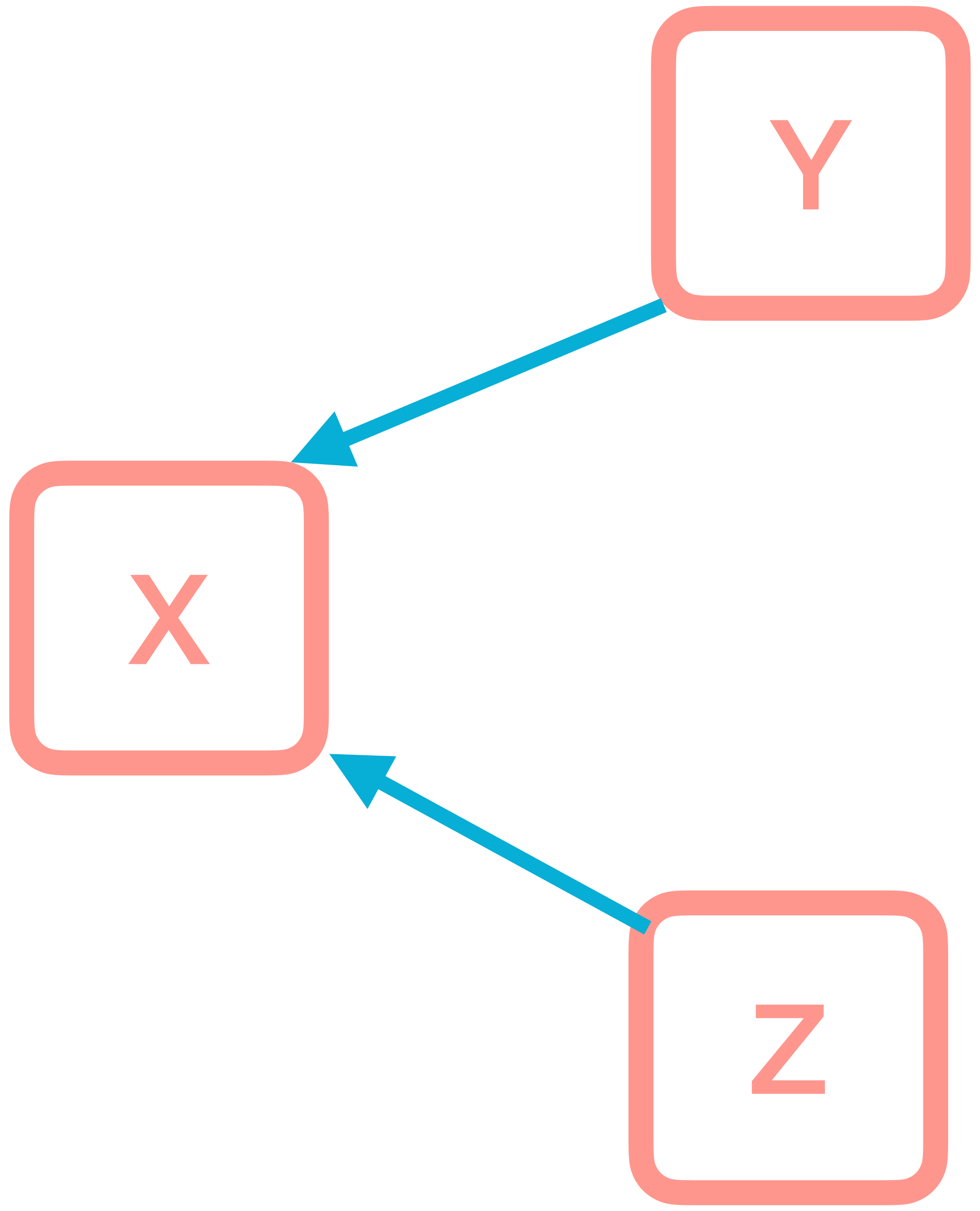
$C = Z_1\,Z_2$



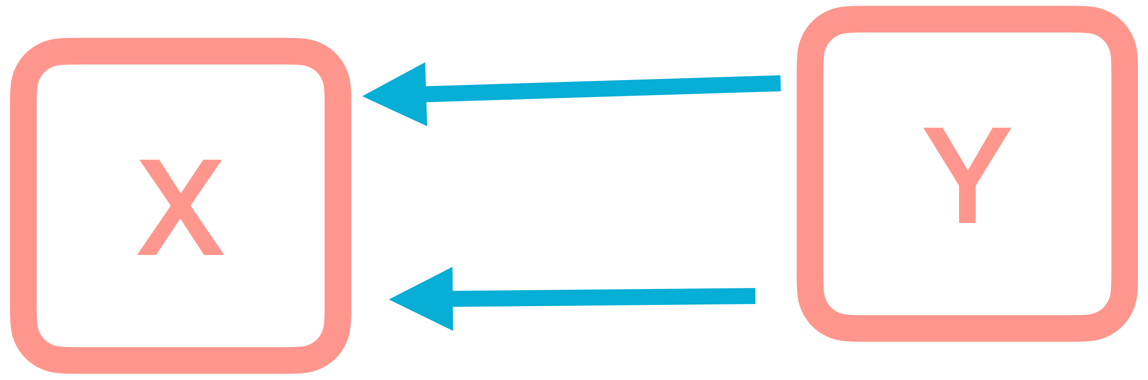
Graph has cycles

Comparison with Tree Fragment

Straight-line is more general than the tree property



X = Y Z (Straight-line and tree)

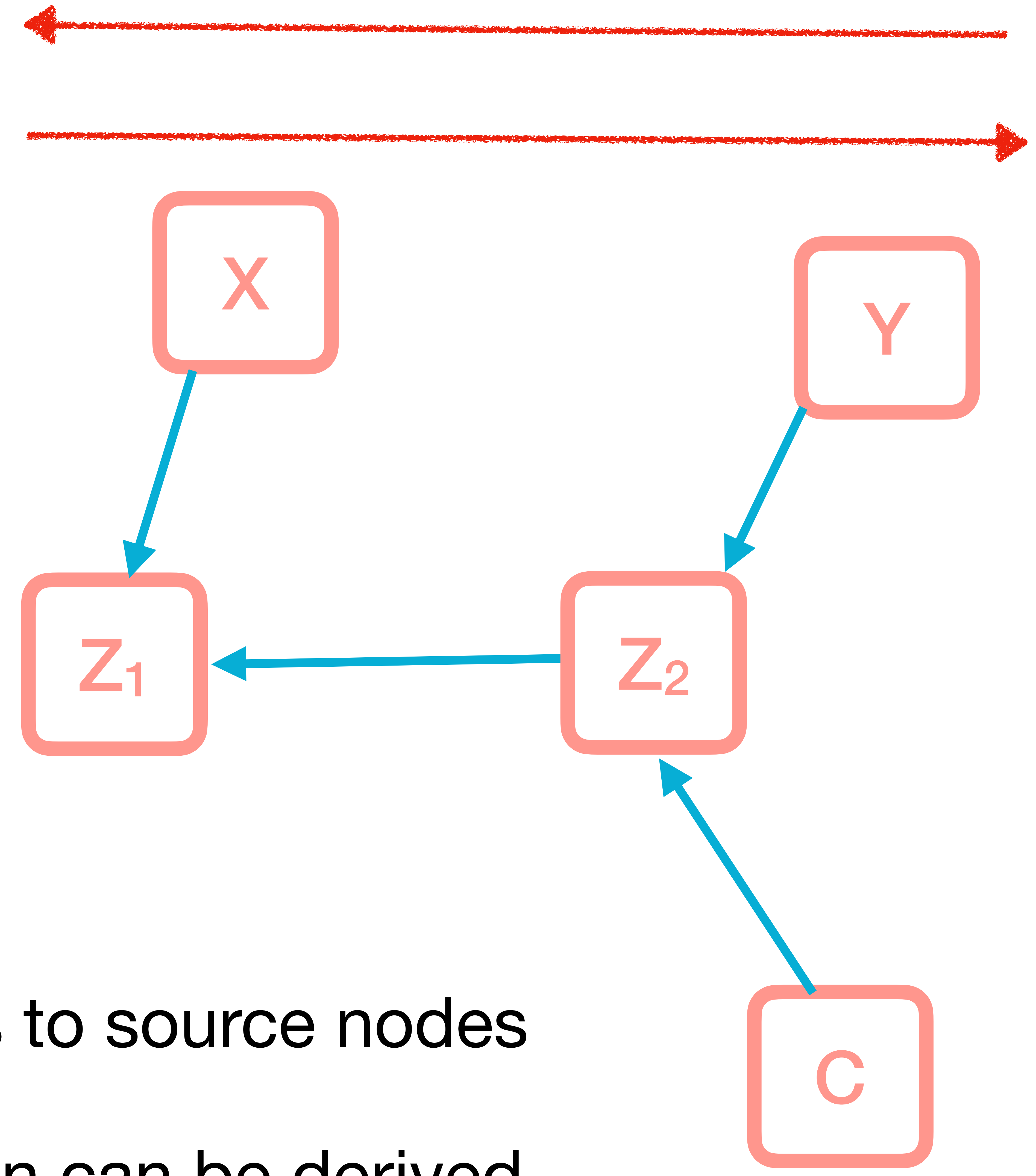


X = Y Y (straight-line, not tree)

Proof Strategy

$$Z_1 = X\, Z_2, \quad Z_2 = C\, Y,$$

$$C \in c, \quad X \in ab^*c, \quad Z_1 \in b\ .^*$$



Start at sink nodes

- Propagate constraints backwards to source nodes
- If no contradictions found, solution can be derived
- Terminates because no cycles

Branching

Backwards propagation creates branching

Each branch has to be explored for unsat

For sat, only one branch needs to succeed

$X \in \varepsilon, \quad Y \in ac^*d^*$

$X \in a, \quad Y \in c^*d^*$

$X \in ac^*, \quad Y \in c^*d^*$

$X \in ac^*, \quad Y \in d^*$

$X \in ac^*d^*, \quad Y \in \varepsilon$

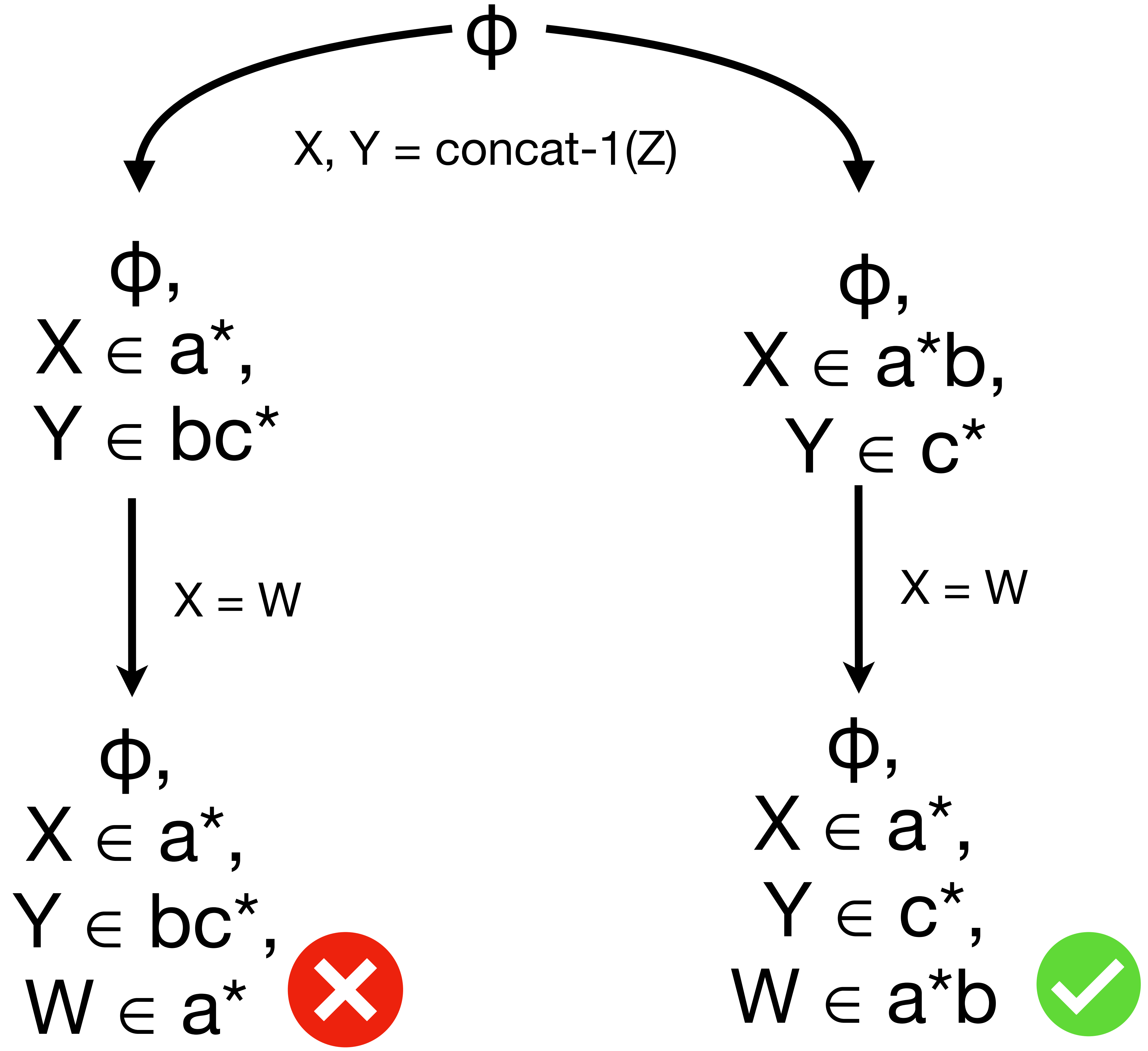
$XY \in ac^*d^*$

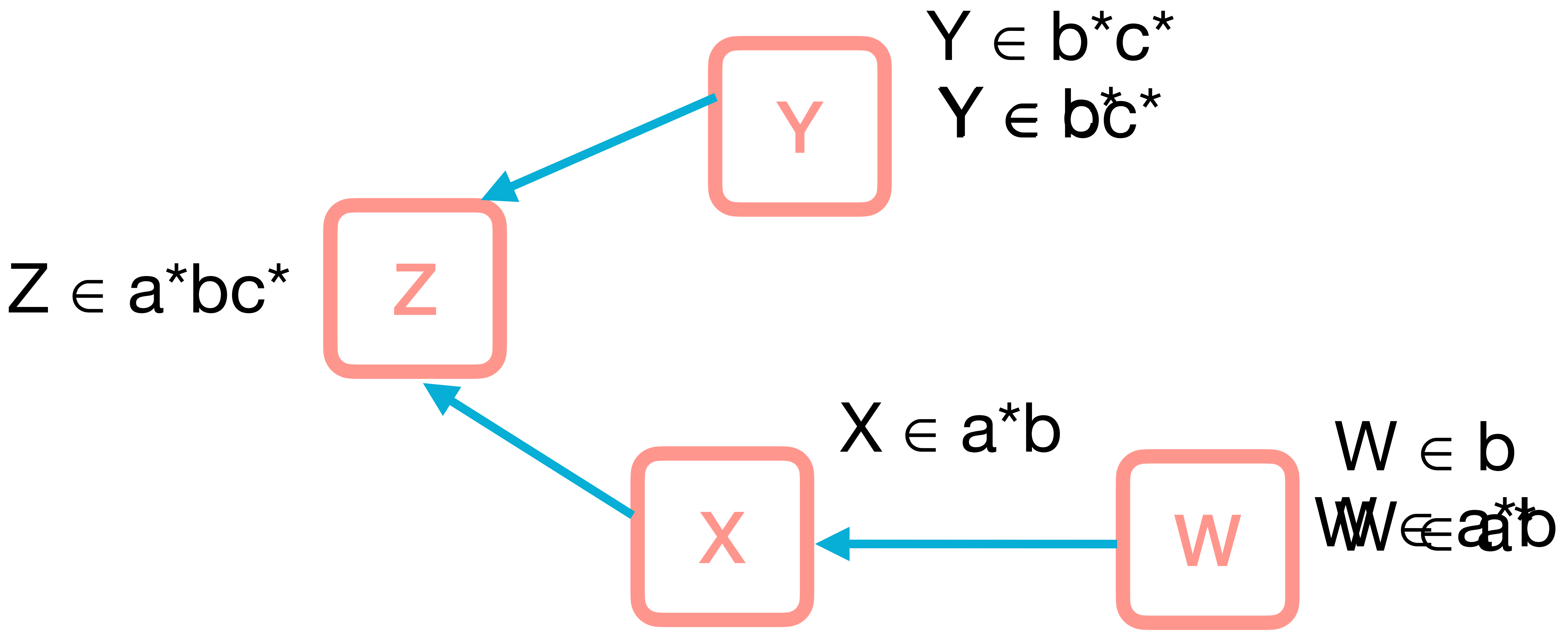
Example

$\phi := W \in b, \quad Y \in b^*c^*, \quad Z \in a^*bc^*,$

$\boxed{X = W,} \boxed{Z = X\,Y}$

Search tree:



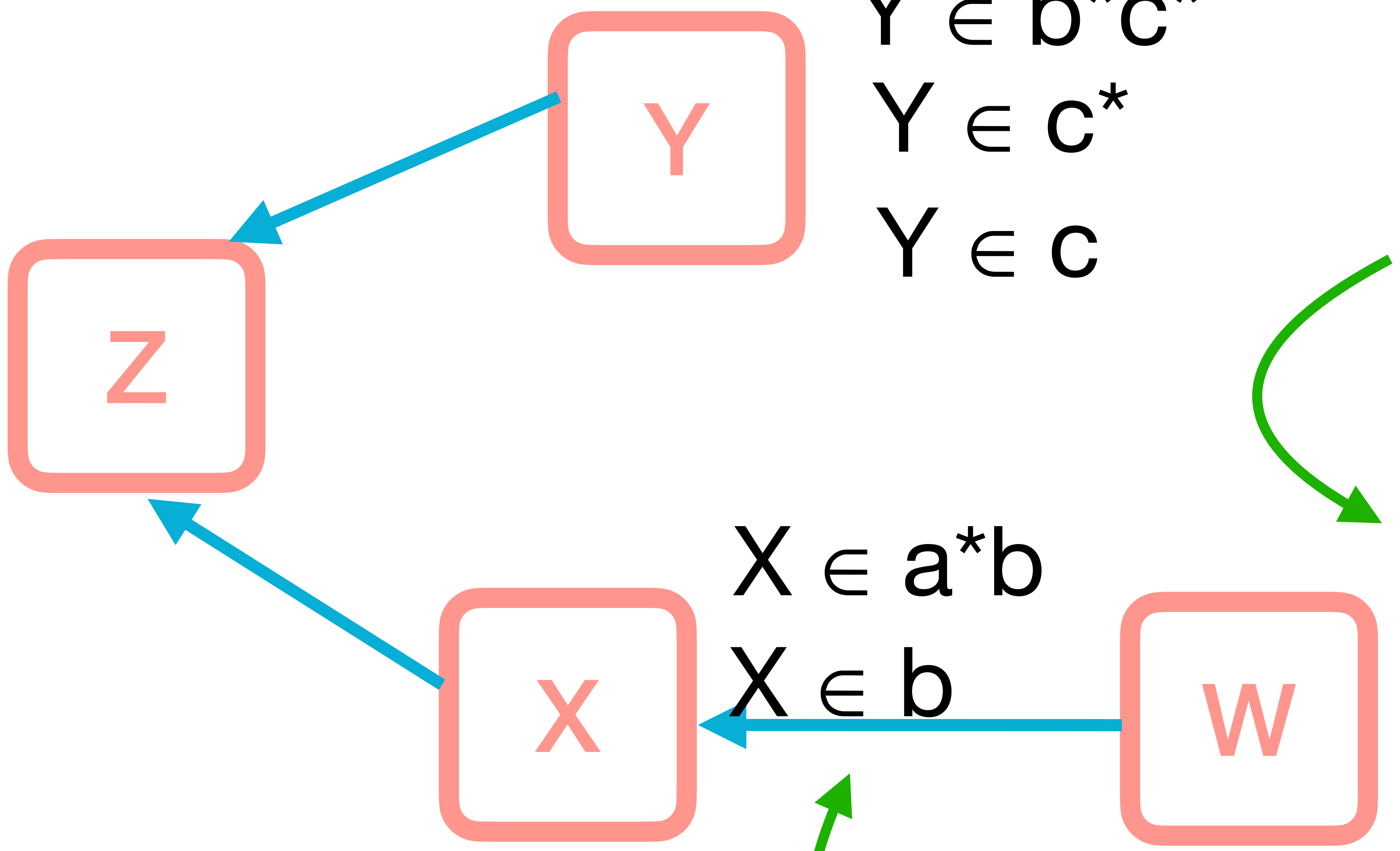


Deriving Solutions

To derive solutions, we start at the source nodes

Eliminate via intersection
Pick value for source node
c ∩ c* ∩ b*c* = c

Eliminate via intersection
Eliminate Z = X Y because Z ∈ b c



Y ∈ b*c*
Y ∈ c*
Y ∈ c

Do the same here

Z ∈ a*bc*
Z ∈ b c

X ∈ a*b
X ∈ b

W ∈ a*b
W ∈ b

Eliminate X = W because one value
Eliminate via intersection

Experimental Evaluation

Initial experiments on propagation strategies on QF_S from SMT-COMP

1. Only backward propagation: T/O on 20 (out of ~7k benchmarks)
2. Only forward propagation: T/O on 33
3. Forward then backward: T/O on 18

OSTRICH: portfolio approach supporting richer constraints

Summary

- Backwards and forwards propagation of constraints
- Solution extraction via cut
- Can support a range of string functions
- Sound, but not always complete
- Completeness for fragments:
  - Tree property
  - Straight-Line
  - Chain-free