Extensions:
Handling of Length and Integers

Overview

- Constraints mixing strings and integers are extremely common
  - In **theory**, very challenging combination
  - In **practice**, there are many fragments that can be solved **completely**
- We introduce three of them:
  1. Straight-line + length constraints that are **monadically decomposable**
  2. Straight-line + **general length** constraints
  3. Straight-line + **general operations** with integers

Strings + Integers

- Word length: $|x|$ `(str.len x)`
- Substring: $x[l..u]$ `(str.substr x l n)`
- Character access: x_i `(str.at x i)`

- Letter counting: count occurrences of ‘a’, *etc.*
- String-to-number conversions:

`(str.from_int d)` `(str.to_int x)`

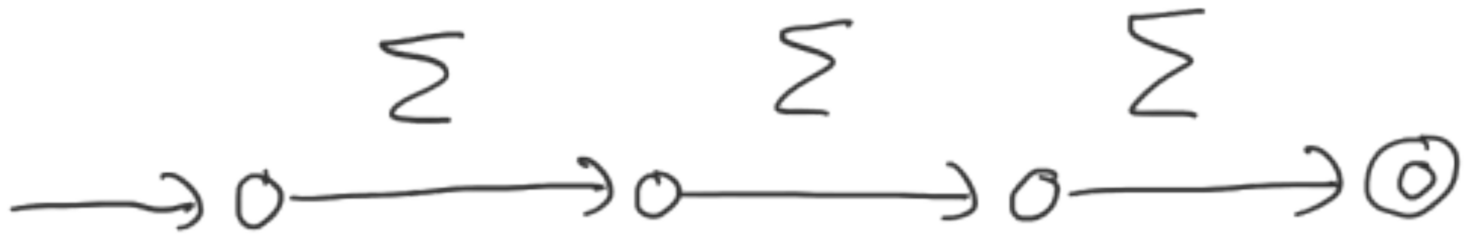
Length constraints

- How can we solve this constraint?

$$x \in L((a + b)^*) \land |x| = 3$$

- **Easy!** The length constraint is equivalent to:

$$x \in \Sigma\Sigma\Sigma$$



- **More generally:** every semi-linear length constraints over **a single string variable** can be replaced with a regular expression

How about the case of multiple variables?

$$|x| = |y| \wedge |x| \leq 3$$

$$|x| = |y|$$

Monadic formulas

- A formula is **monadic** if it is a Boolean combination of monadic predicates (at most one variable per predicate).
- A formula is **monadically decomposable** if it is equivalent to a monadic formula.

- **Observation:** if a formula in variables $|x_1|, \ldots, |x_n|$ is monadically decomposable, then it describes a **recognisable relation** between $x_1, \ldots, x_n$

Margus Veanes, Nikolaj Bjørner, Lev Nachmanson, Sergey Bereg:
Monadic decomposition. J. ACM 64(2), 14:1–14:28 (2017).

Monadic decomposition in benchmarks

- Experiments with the Kaluza benchmarks: only **4713 out of 47284** problems contained relevant non-monadic length constraints

Folder	#Benchmarks	Benchmarks with `str.len`	Decomposition checks	Decomposition checks succeeded
sat/small	19804	2185	2183	2155
sat/big	1741	1318	1317	56
unsat/small	11365	3910	2919	2919
unsat/big	14374	13813	6786	3362
Total	47284	21226	13205	8492

Matthew Hague, Anthony W. Lin, Philipp Rümmer, Zhilin Wu:
Monadic Decomposition in Integer Linear Arithmetic. IJCAR 2020

Different fragments with integers

1. Straight-line, all length constraints monadically decomposable
   - Can directly be handled in the propagation-based framework ✅

2. Straight-line, general length constraints

3. Straight-line, general operations with integers

General length constraints

$$x \in L_1 \wedge y \in L_2 \wedge |x| = 2 \cdot |y|$$

Find formula $\phi_1[\,|x|\,]$ representing lengths of all included words

$\phi_2[\,|y|\,]$

- **Solution:** represent both length constraints and regular languages using **arithmetic formulas**

- The original formula is sat if and only if the **length abstraction** is sat:

$$\phi_{L_1}(|x|) \wedge \phi_{L_2}(|y|) \wedge |x| = 2 \cdot |y|$$

Length abstractions of regular languages

- The length abstraction is a special case of the **Parikh image**
- An existential Presburger formula representing the length abstraction can be computed in linear time

Kumar Neeraj Verma, Helmut Seidl, Thomas Schwentick:
On the Complexity of Equational Horn Clauses. CADE 2005: 337-352

Different fragments with integers

1. Straight-line, all length constraints monadically decomposable ✅
   - Can directly be handled in the propagation-based framework
2. Straight-line, general length constraints ✅
   - Supports only certain functions: concat, length-preserving transducers
   - Approach introduced in the SMT solver **Norn**
3. Straight-line, general operations with integers

Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Lukás Holík, Ahmed Rezine, Philipp Rümmer, Jari Stenman: **String Constraints for Verification**. CAV 2014

General constraints involving integers

- How can we solve this constraint?

$$y = x[l\,..\,r] \wedge x \in L_1 \wedge y \in L_2 \wedge r \geq 2 \cdot l$$

Perform backwards-propagation
to derive a cost-enriched constraint
$(x, l, r) \in L_2'$

Check consistency
of the cost-enriched
constraints

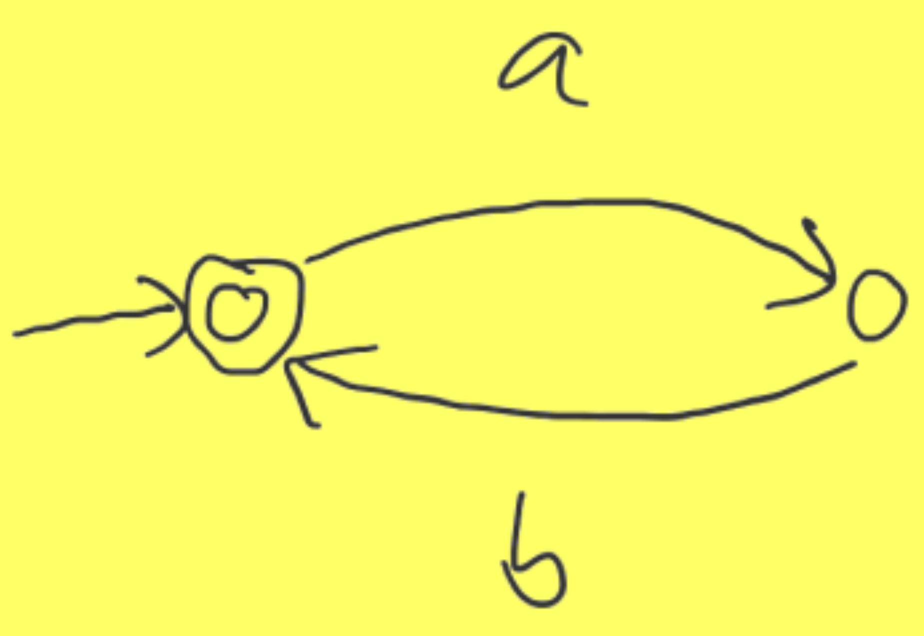
- **Solution:** tightly integrate length + regular constraints:
  Cost-enriched automata (CEA)

Taolue Chen, Matthew Hague, Jinlong He, Denghang Hu, Anthony W. Lin, Philipp Rümmer, Zhilin Wu: **A Decision Procedure for Path Feasibility of String Manipulating Programs with Integer Data Type.** ATVA 2020

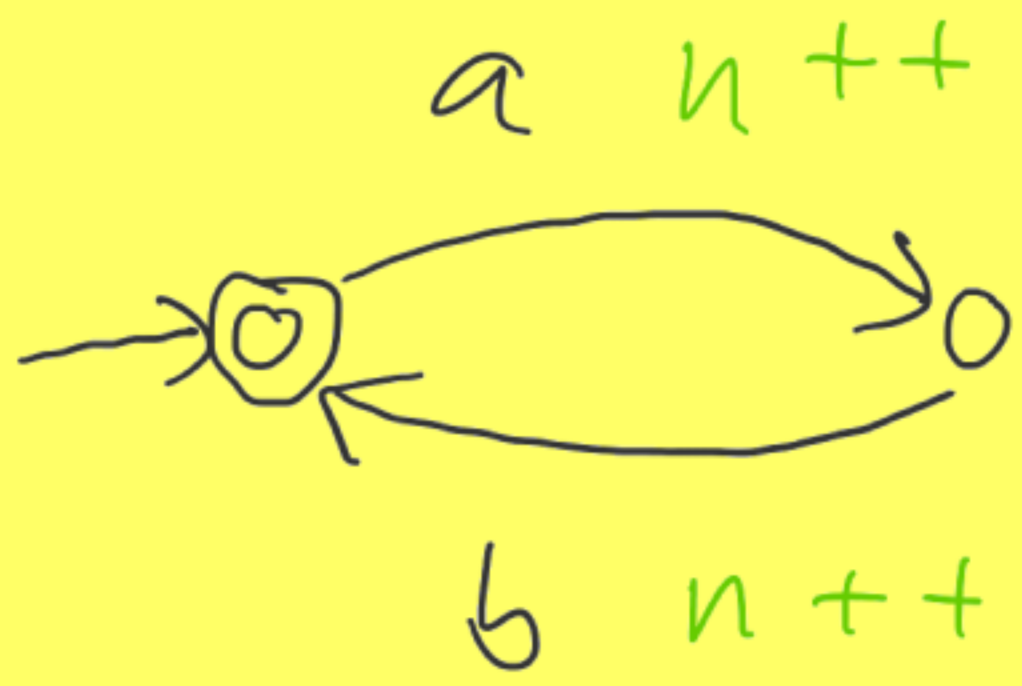
Cost-enriched automata (CEA)

- Automata augmented with counters
  - Counters start at zero
  - Transitions can increment or decrement counters
  - No zero-tests

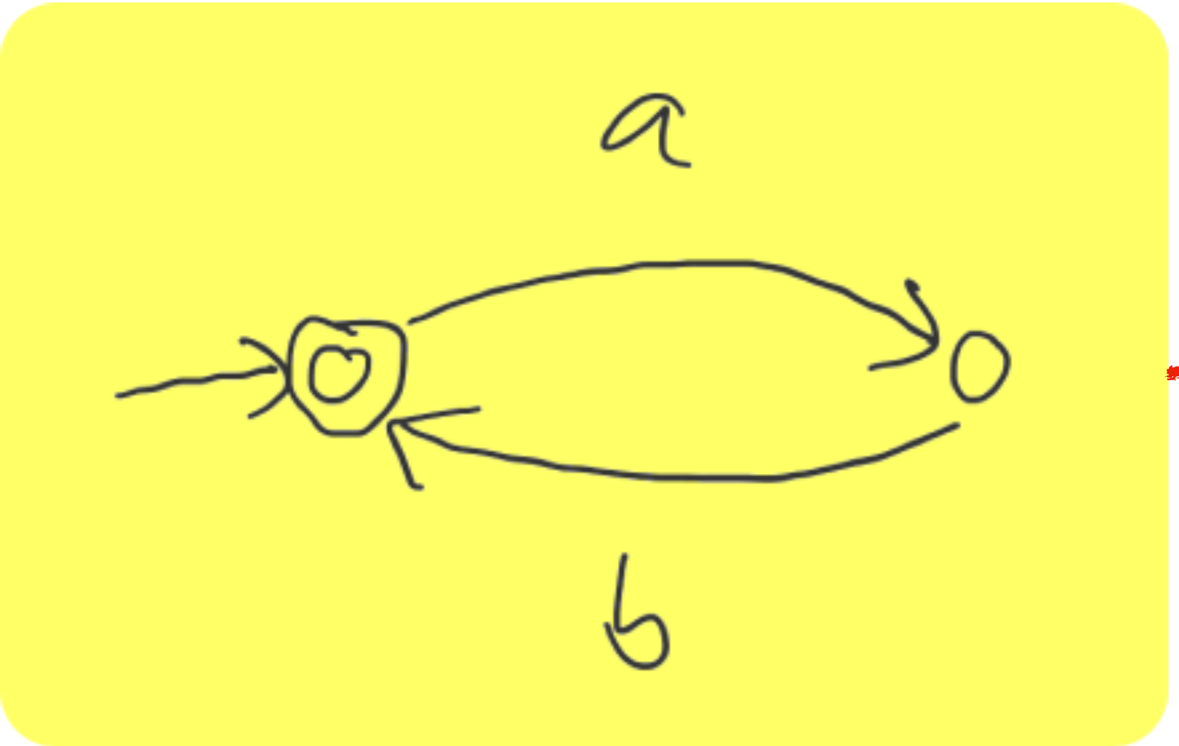
$$x \in (ab)*$$



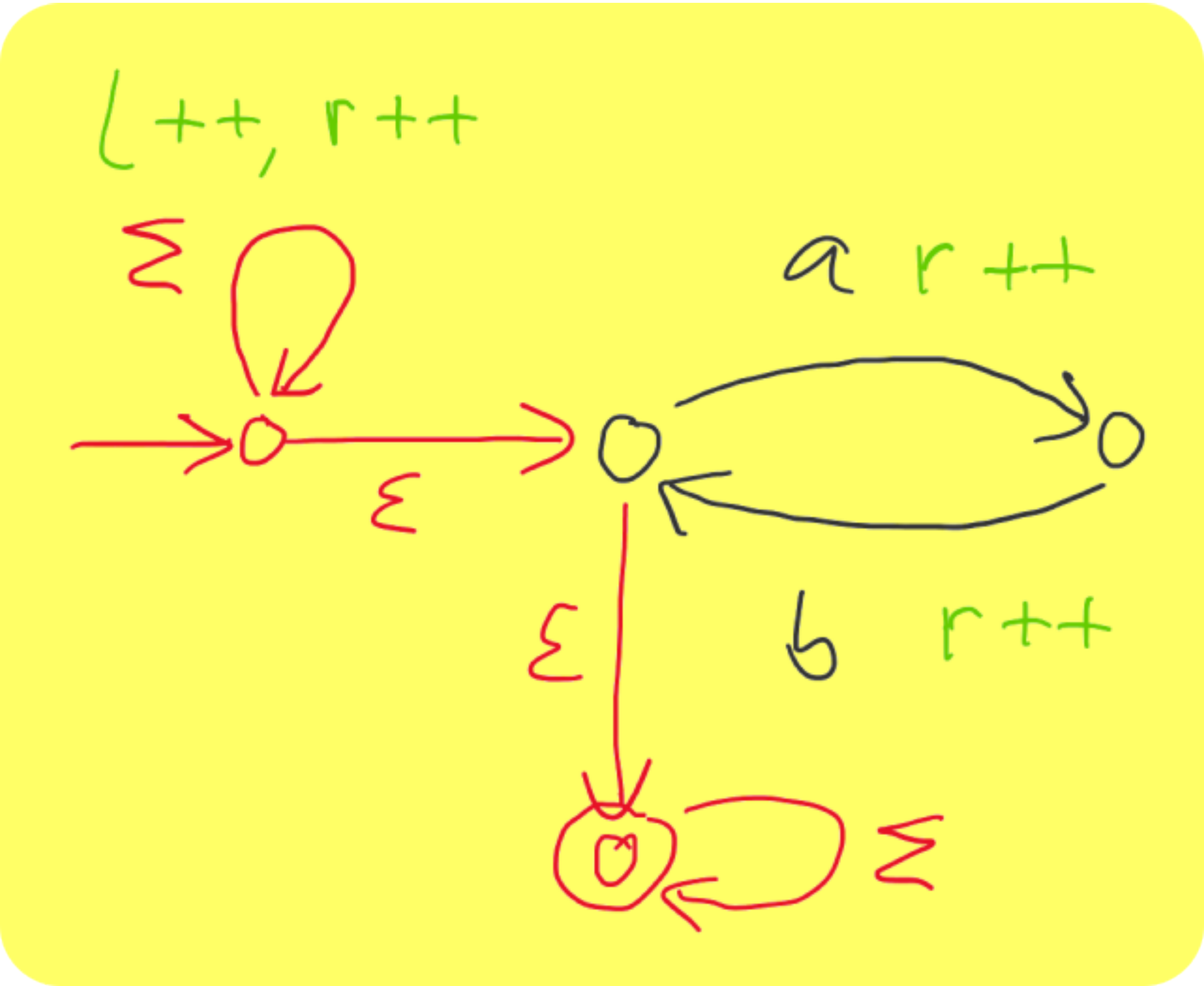
$$(x, n) \in \{(w, |w|) \mid w \in (ab)*\}$$



Backwards-propagation with CEA



$y = x[l\,..\,r]$



- This works for: concat, substring, replace-all, reverse, *etc.*

CEA consistency checking

- After propagation:
  - Compute the products of all CEAs for the same string variable
  - Reachable counter values are extracted from Parikh image of the product
- Relatively expensive → Use **laziness** to speed up the checks

Amanda Stjerna, Philipp Rümmer: **A Constraint Solving Approach to Parikh Images of Regular Languages.** OOPSLA 2024

CEA backend in OSTRICH

- At the moment, the CEA back-end is separate from the standard automata back-end
    - On the web interface: menu to choose the back-end to apply