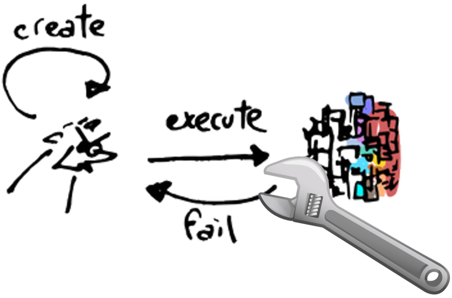
THE COST OF REPAIRS



Gabriele Puppis

LaBRI / CNRS

based on joint works with

Michael Benedikt,
Pierre Bourhis,
Cristian Riveros,
Slawek Staworko

What do you do when a computational object fails a specification?

Source

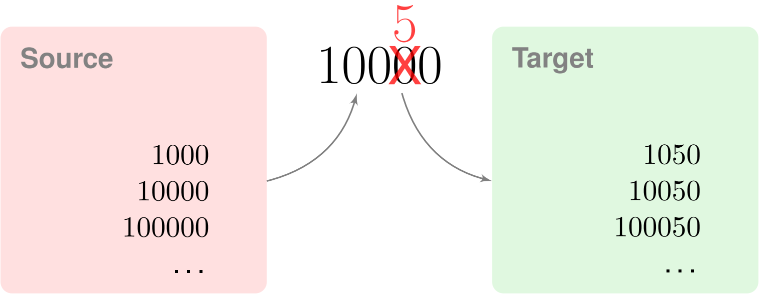
1000
10000
100000
$\dots$

$$100\overset{5}{\cancel{0}}0$$

Target

1050
10050
100050
$\dots$

What do you do when a computational object fails a specification?



Worst-case cost of repairing source into target:

$$\max_{s \in S} \min_{t \in T} \mathsf{dist}(s, t)$$

What do you do when a computational object fails a specification?

Source

1000
10000
100000
$\dots$

$$100\overset{5}{\cancel{0}}0$$

Target

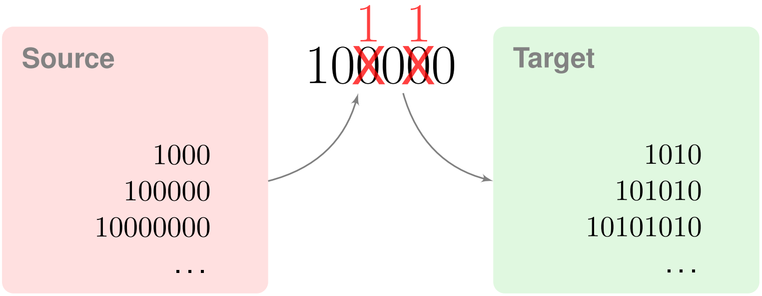
1050
10050
100050
$\dots$

Worst-case cost of repairing source into target:

$$\max_{s \in S} \min_{t \in T} \mathsf{dist}(s, t)$$

Depends on distance (e.g., Levenstein distance)

What do you do when a computational object fails a specification?



Worst-case cost of repairing source into target:

$$\max_{s \in S} \min_{t \in T} \mathsf{dist}(s, t)$$

- Depends on distance (e.g., Levenstein distance)
- Can be **infinite**!

Plan

A. **Bounded repairability of regular word languages**

1) characterization

2) streaming setting

3) complexity

B. **Bounded repairability of regular tree languages**

1) curry encodings, stepwise automata, contexts

2) characterization

3) complexity

Part A. Problem setting:

- Given two languages $S \subseteq \Sigma^*$ and $T \subseteq \Delta^*$
  (represented by finite state automata)

- Decide whether $\max_{s \in S} \min_{t \in T} \mathrm{dist}(s, t)$ is finite.

Part A. Problem setting:

- Given two languages $S \subseteq \Sigma^*$ and $T \subseteq \Delta^*$
  (represented by finite state automata)

- Decide whether $\max_{s \in S} \min_{t \in T} \mathrm{dist}(s,t)$ is finite.

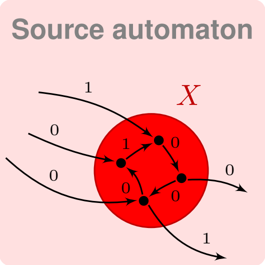
Examples

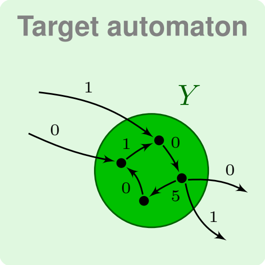
10^* is <u>bounded repairable</u> into 10^*50

10^* is <u>not</u> bounded repairable into $(10)^*$

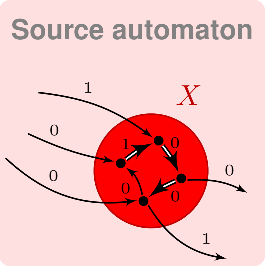
$(1+0)^*$ is <u>not</u> bounded repairable into $(1+0^*5)^*$

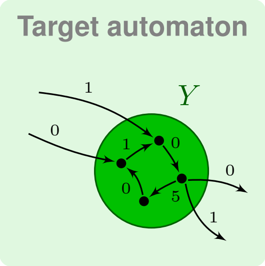
Rule of thumb: *“If you need to edit, you’d better do it outside a loop!”*



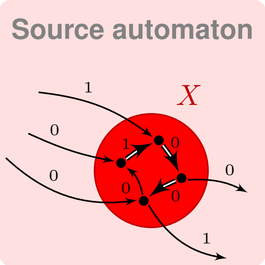


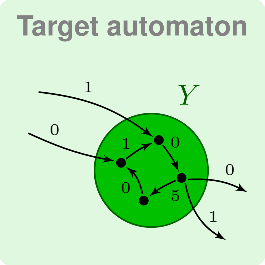
Rule of thumb: *“If you need to edit, you’d better do it outside a loop!”*





Rule of thumb: “*If you need to edit, you'd better do it outside a loop!*”





For any strategy that repairs traces of X into traces of Y:

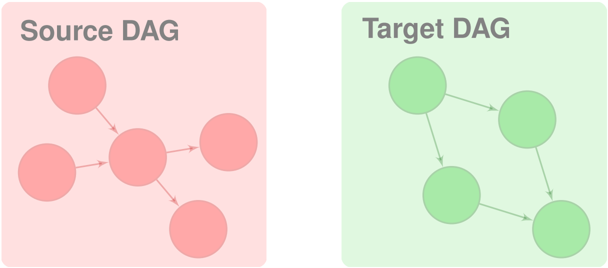
1. either $\mathsf{traces}(X) \subseteq \mathsf{traces}(Y)$
2. or the strategy has unbounded cost.

Characterization of bounded repairability of word languages

S is repairable into T with uniformly bounded cost

$$\Updownarrow$$

Given some (trimmed) automata for S and T
and the DAGs of strongly connected components...

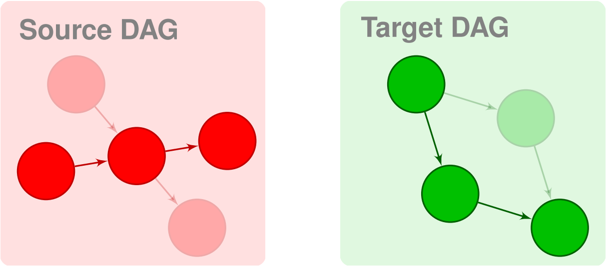


Characterization of bounded repairability of word languages

S is repairable into T with uniformly bounded cost

$$\Updownarrow$$

Given some (trimmed) automata for S and T
and the DAGs of strongly connected components...



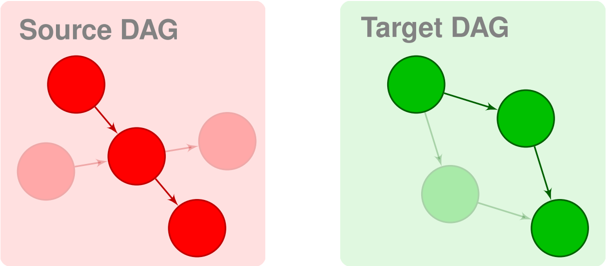
...every chain of components in the source is
covered by a chain of components in the target.

Characterization of bounded repairability of word languages

S is repairable into T with uniformly bounded cost

$$\Updownarrow$$

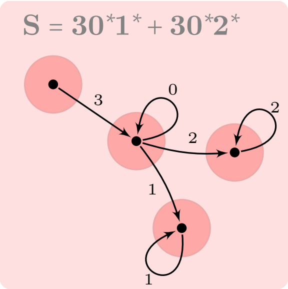
Given some (trimmed) automata for S and T
and the DAGs of strongly connected components...

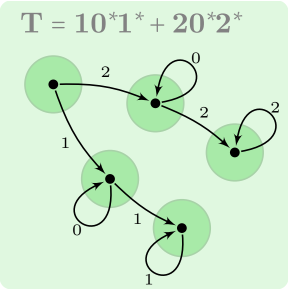


...every chain of components in the source is
covered by a chain of components in the target.

An example

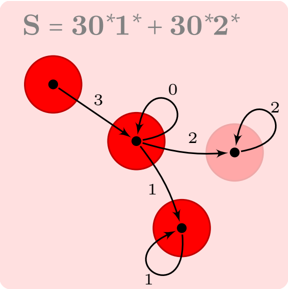
All chains of source DAG are covered by chains of target DAG
$\Rightarrow$ $\color{red}{S}$ is repairable into $\color{green}{T}$ with uniformly bounded cost.

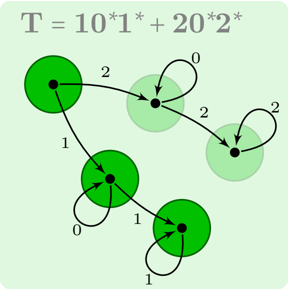




An example

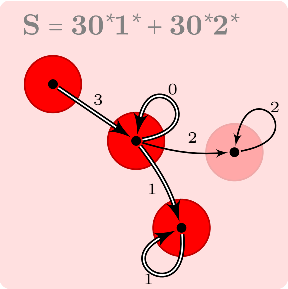
All chains of source DAG are covered by chains of target DAG
$\Rightarrow$ S is repairable into T with uniformly bounded cost.

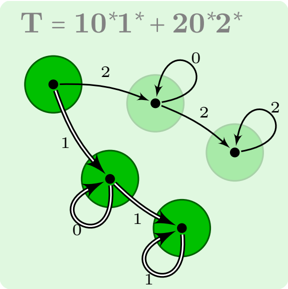




An example

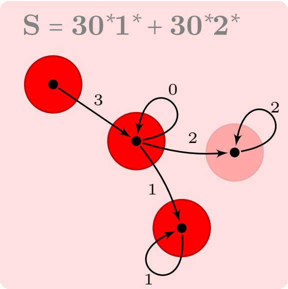
All chains of source DAG are covered by chains of target DAG
$\Rightarrow$ S is repairable into T with uniformly bounded cost.

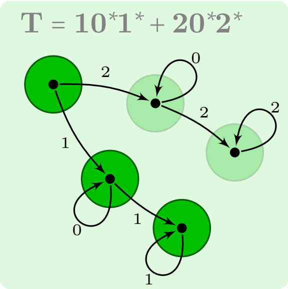




An example

All chains of source DAG are covered by chains of target DAG
$\Rightarrow$ S is repairable into T with uniformly bounded cost.



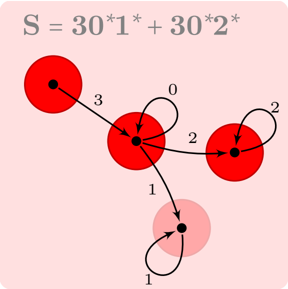


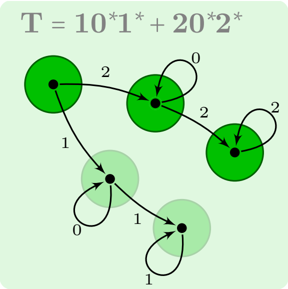
There is no covering relation **compatible with prefixes**
$\Rightarrow$ the repair strategy is not **streaming**
(i.e. implementable by a sequential transducer)

An example

All chains of source DAG are covered by chains of target DAG

⇒ S is repairable into T with uniformly bounded cost.





There is no covering relation **compatible with prefixes**

⇒ the repair strategy is not **streaming**

(i.e. implementable by a sequential transducer)

Complexity of **non-streaming** bounded repairability problem:

	fixed	DFA	NFA
fixed	**CONST**	**P**	**PSPACE**
DFA	**P**	**coNP**	**PSPACE**
NFA	**PTIME**	**coNP**	**PSPACE**

Complexity of **non-streaming** bounded repairability problem:

	fixed	DFA	NFA
fixed	**CONST**	**P**	**PSPACE**
DFA	**P**	**coNP**	**PSPACE**
NFA	**PTIME**	**coNP**	**PSPACE**

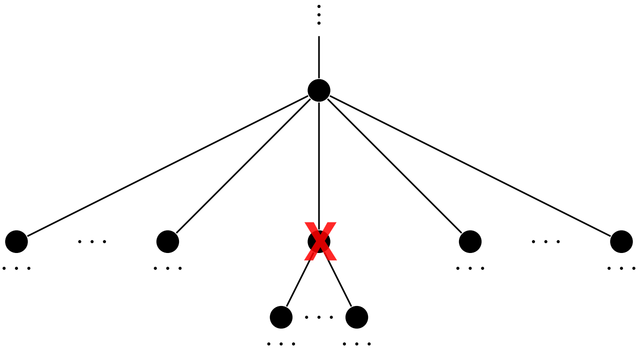
Complexity of **streaming** bounded repairability problem:

	fixed	DFA	NFA
fixed	**CONST**	**P**	**PSPACE**
DFA	**P**	**P**	**PSPACE**
NFA	$\leq$ **PSPACE** $\geq$ **P**	$\leq$ **PSPACE** $\geq$ **P**	$\leq$ **EXP** $\geq$ **PSPACE**

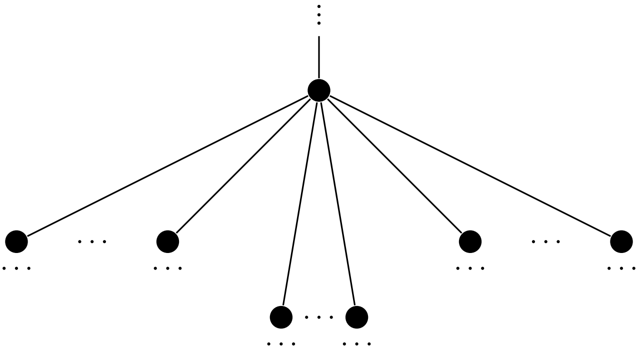
Part B. New tools for a more general setting...

Languages of words:	Languages of unranked trees:
insersions / deletions	**insertions / deletions**
finite state automata	**stepwise tree automata**
components & traces	**components & contexts**
coverability of chains	**coverability of synopsis trees**

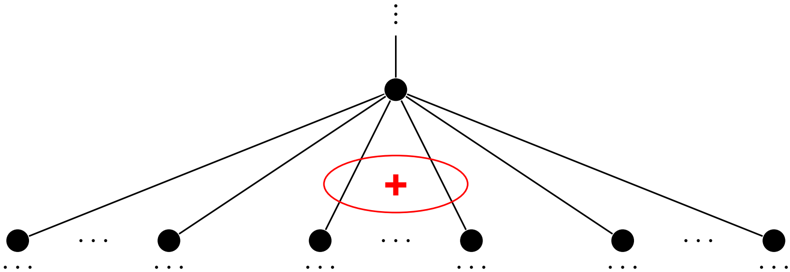
Edit operations on unranked trees: **deletions**



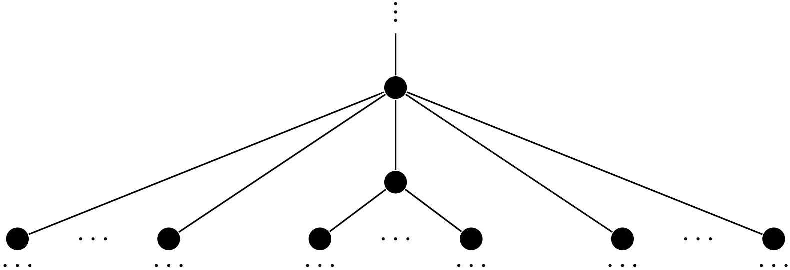
Edit operations on unranked trees: **deletions**



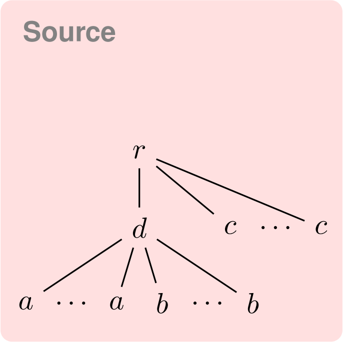
Edit operations on unranked trees: **insertions**

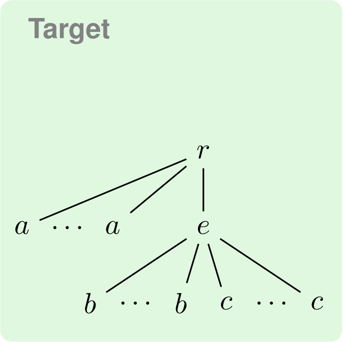


Edit operations on unranked trees: **insertions**

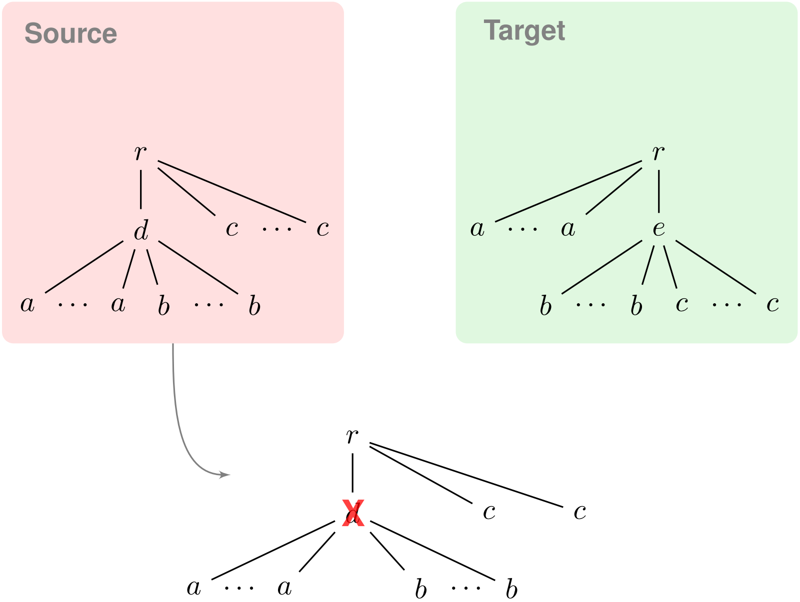


An example



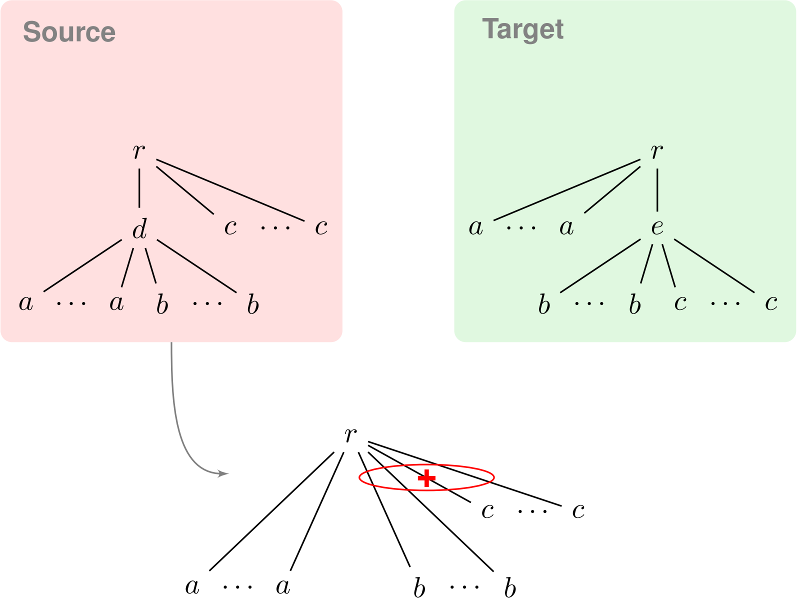


An example

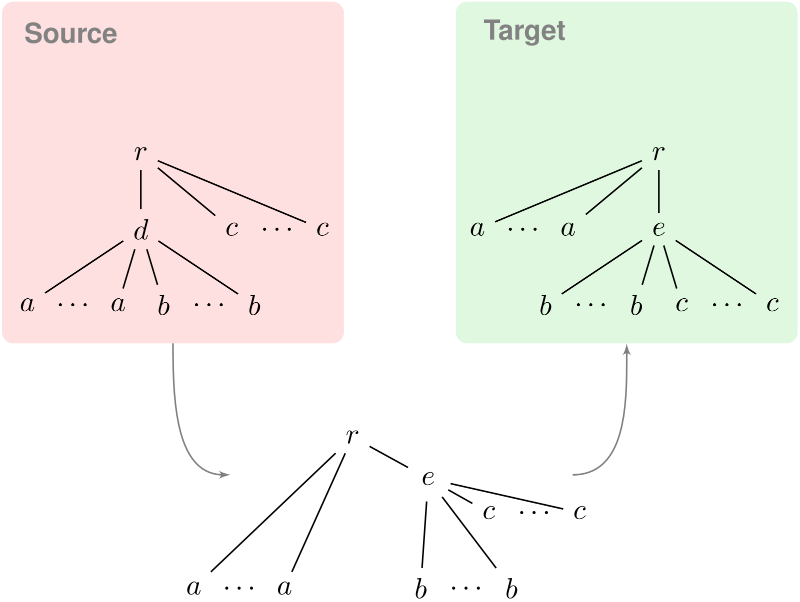


Source
r
d
$c \cdots c$
$a \cdots a$
$b \cdots b$
Target
r
$a \cdots a$
e
$b \cdots b$
$c \cdots c$
r
d
c
c
$a \cdots a$
$b \cdots b$

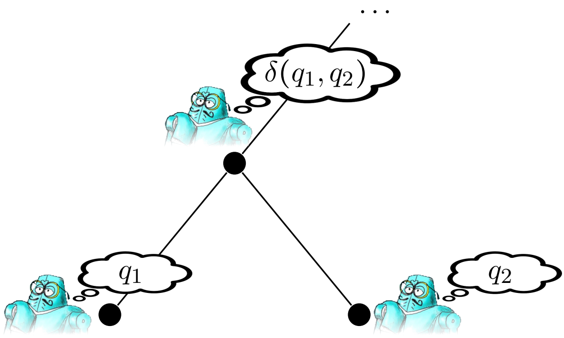
An example



An example

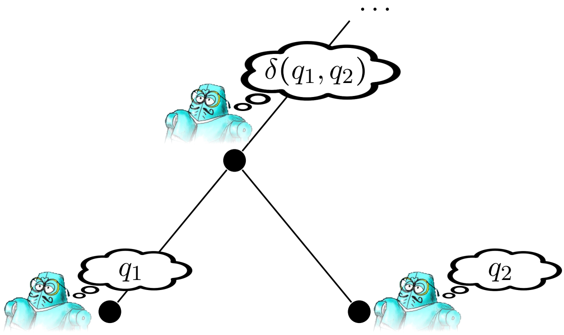


Bottom-up automata on ranked (binary) trees:



How to parse unranked trees?

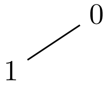
Bottom-up automata on ranked (binary) trees:



How to parse unranked trees?

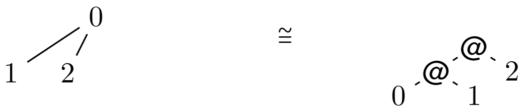
☞ Encode them using binary trees!

The curry encoding

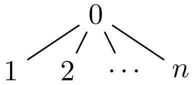


$\cong$

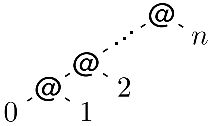
The curry encoding



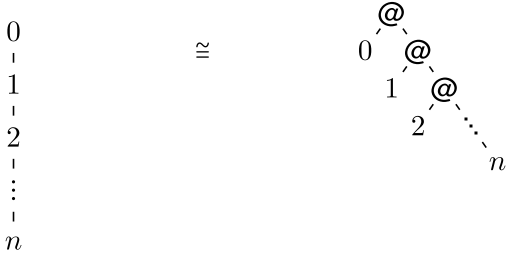
The curry encoding



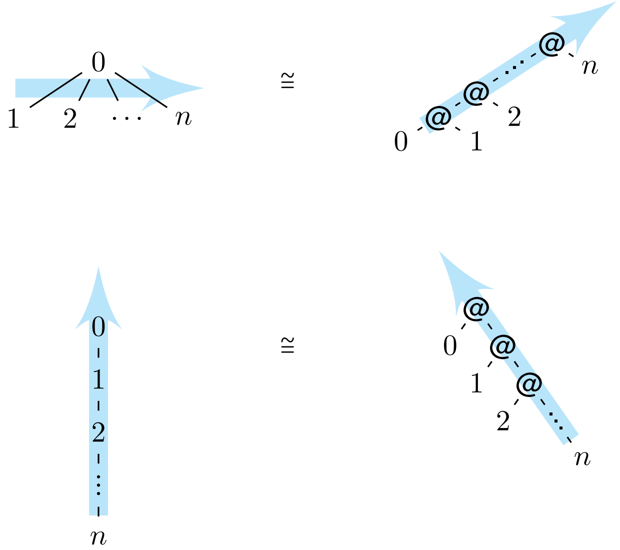
$\cong$



The curry encoding

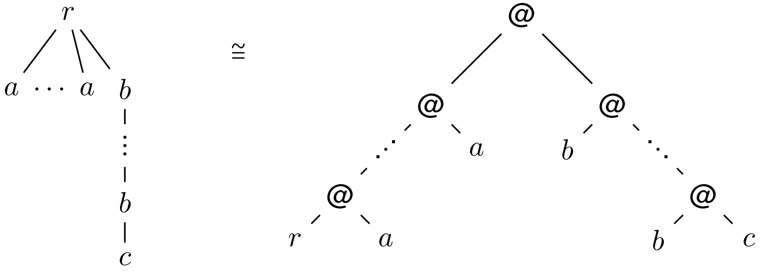


The curry encoding

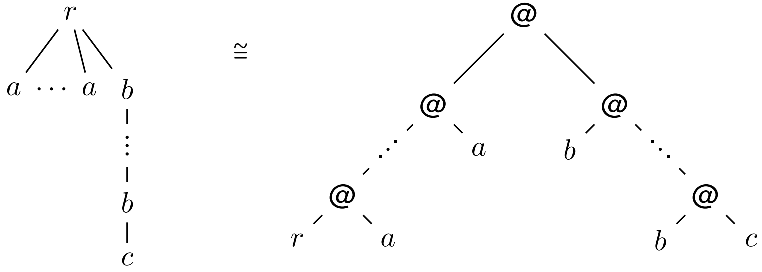


☞ Stepwise automata = bottom-up on curry encodings

An example



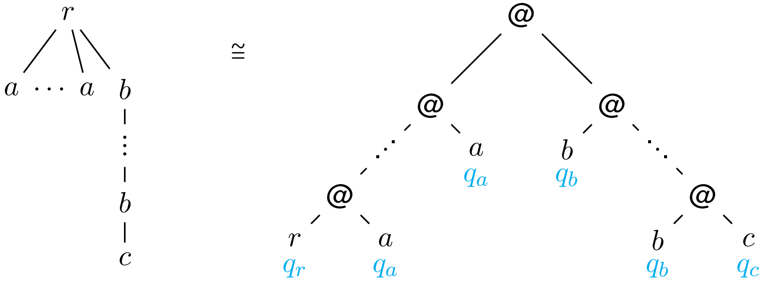
An example



$$\begin{array}{ccc} r & \mapsto & q_r \\ a & \mapsto & q_a \\ b & \mapsto & q_b \\ c & \mapsto & q_c \end{array}$$

$$\begin{array}{ccc} q_r \ @ \ q_a & \mapsto & q_r \\ q_b \ @ \ q_c & \mapsto & q_c \\ q_r \ @ \ q_c & \mapsto & q_{\mathsf{final}} \end{array}$$

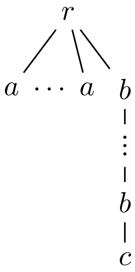
An example



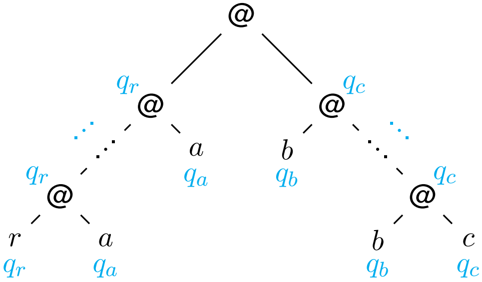
$$\begin{array}{ccc} r & \mapsto & q_r \\ a & \mapsto & q_a \\ b & \mapsto & q_b \\ c & \mapsto & q_c \end{array}$$

$$\begin{array}{ccc} q_r \mathbin{@} q_a & \mapsto & q_r \\ q_b \mathbin{@} q_c & \mapsto & q_c \\ q_r \mathbin{@} q_c & \mapsto & q_{\mathsf{final}} \end{array}$$

An example

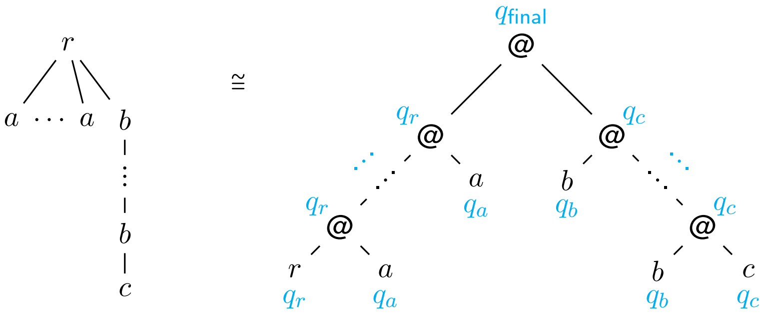


$\cong$



$$\begin{array}{ccc} r & \mapsto & q_r \\ a & \mapsto & q_a \\ b & \mapsto & q_b \\ c & \mapsto & q_c \end{array} \qquad \begin{array}{ccc} q_r \,@\, q_a & \mapsto & q_r \\ q_b \,@\, q_c & \mapsto & q_c \\ q_r \,@\, q_c & \mapsto & q_{\mathsf{final}} \end{array}$$

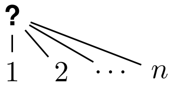
An example



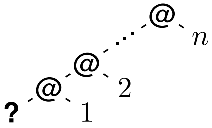
$$\begin{array}{ccc} r & \mapsto & q_r \\ a & \mapsto & q_a \\ b & \mapsto & q_b \\ c & \mapsto & q_c \end{array}$$

$$\begin{array}{ccc} q_r \ @ \ q_a & \mapsto & q_r \\ q_b \ @ \ q_c & \mapsto & q_c \\ q_r \ @ \ q_c & \mapsto & q_{\mathsf{final}} \end{array}$$

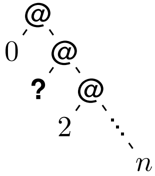
Contexts = trees with holes



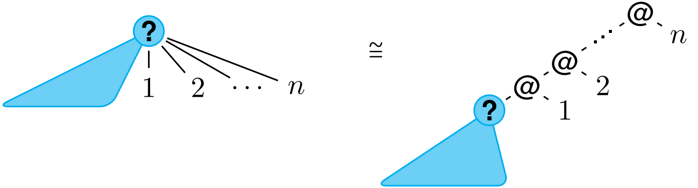
$\cong$

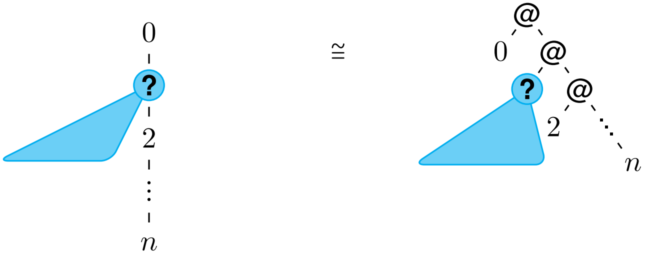


$\cong$

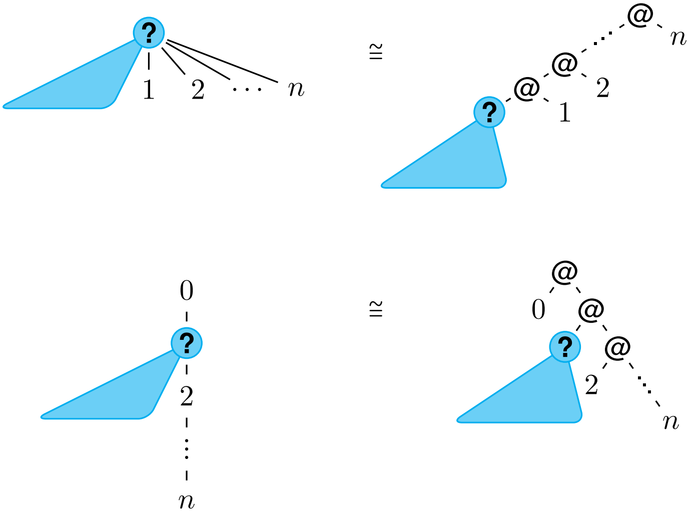


Contexts = trees with holes





Contexts = trees with holes



Contexts can be parsed between two states: $p \xrightarrow{C} q$

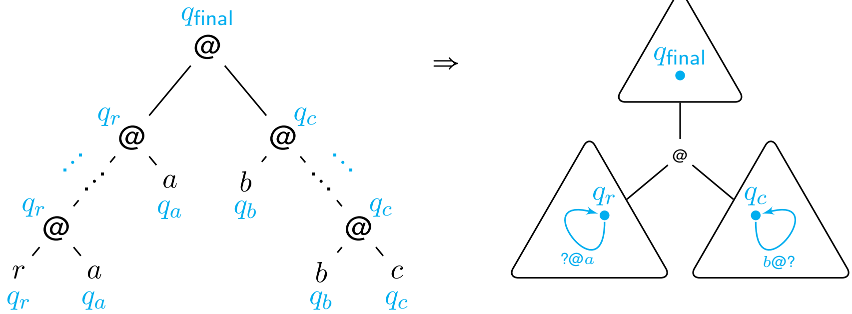
(accessibility of states and components are defined accordingly)

Recall: a run of a finite state automaton induces a **chain of components**...

Likewise, a run of a stepwise automaton induces a **synopsis tree** (i.e. a tree of components).

Recall: a run of a finite state automaton induces a **chain of components**...

Likewise, a run of a stepwise automaton induces a **synopsis tree** (i.e. a tree of components).

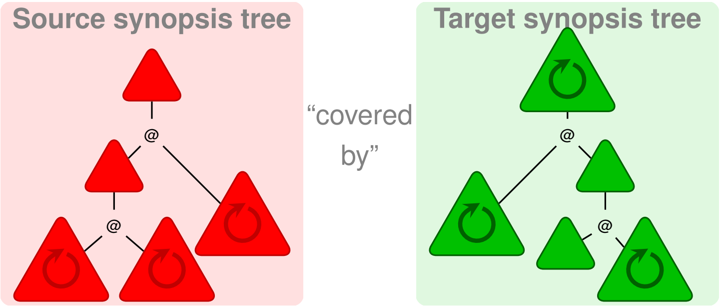


Characterization of bounded repairability of tree languages

S is repairable into T with uniformly bounded cost

$$\Updownarrow$$

Given some (trimmed) stepwise automata for S and T,
all synopsis trees of S are covered by synopsis trees of T



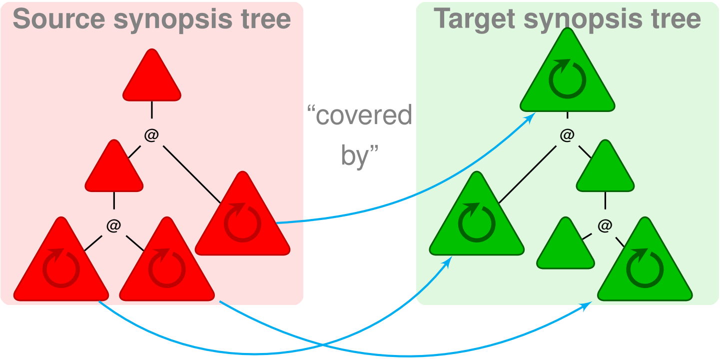
i.e. ...

Characterization of bounded repairability of tree languages

S is repairable into T with uniformly bounded cost

$$\Updownarrow$$

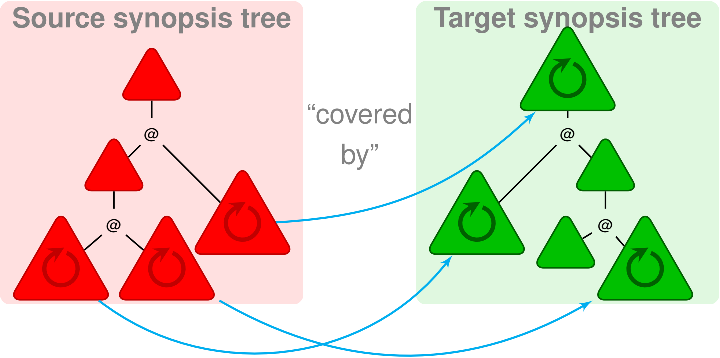
Given some (trimmed) stepwise automata for S and T,
all synopsis trees of S are covered by synopsis trees of T



i.e. $\exists\ \lambda$: cyclic components $\longrightarrow$ cyclic components

$\Updownarrow$

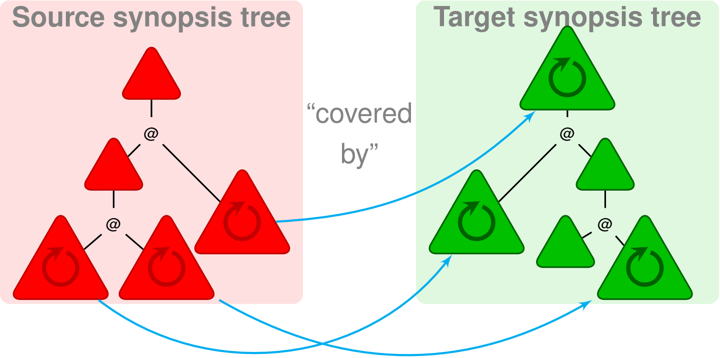
Given some (trimmed) stepwise automata for S and T,
all synopsis trees of S are covered by synopsis trees of T



i.e. $\exists\ \lambda$: cyclic components $\longrightarrow$ cyclic components

1. λ preserves contexts:
contexts(X) $\subseteq$ contexts$(\lambda(X))$

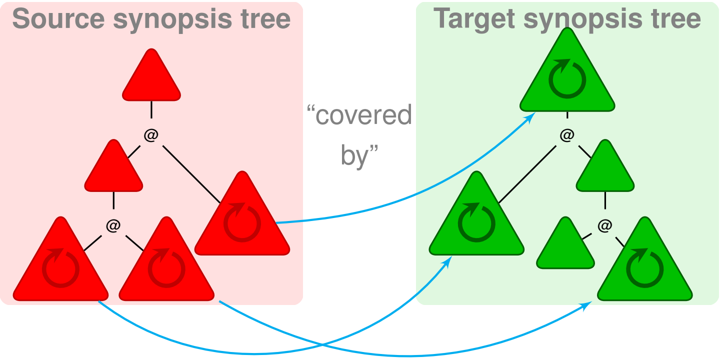
Given some (trimmed) stepwise automata for S and T,
all synopsis trees of S are covered by synopsis trees of T



i.e. $\exists\ \lambda$: cyclic components $\longrightarrow$ cyclic components

1. λ preserves contexts:
   $\text{contexts}(X) \subseteq \text{contexts}(\lambda(X))$

2. λ respects post-order of components:
   $X \leq_{\text{postorder}} Y \leftrightarrow \lambda(X) \leq_{\text{postorder}} \lambda(Y)$

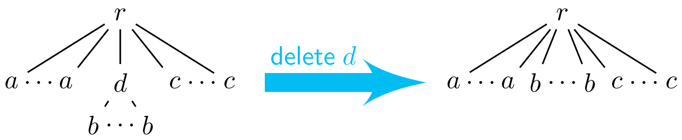


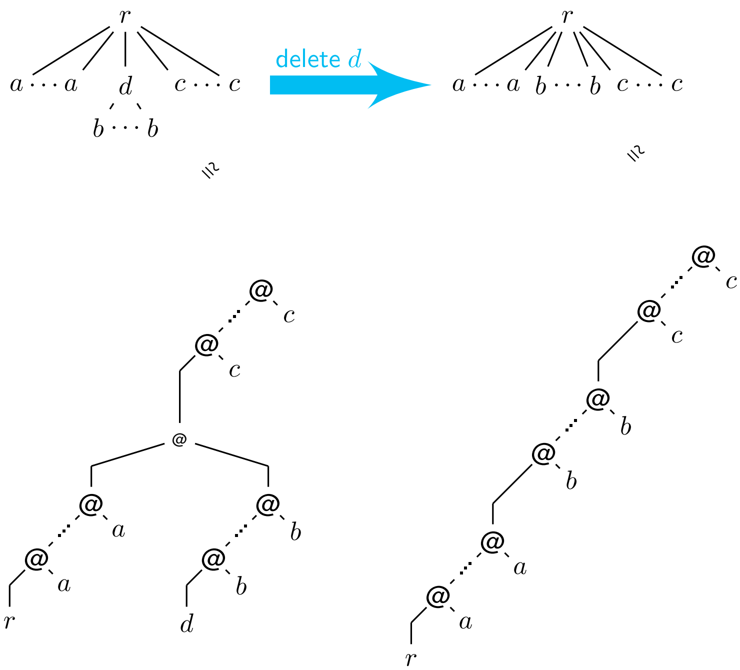
i.e. $\exists\ \lambda$: cyclic components $\longrightarrow$ cyclic components

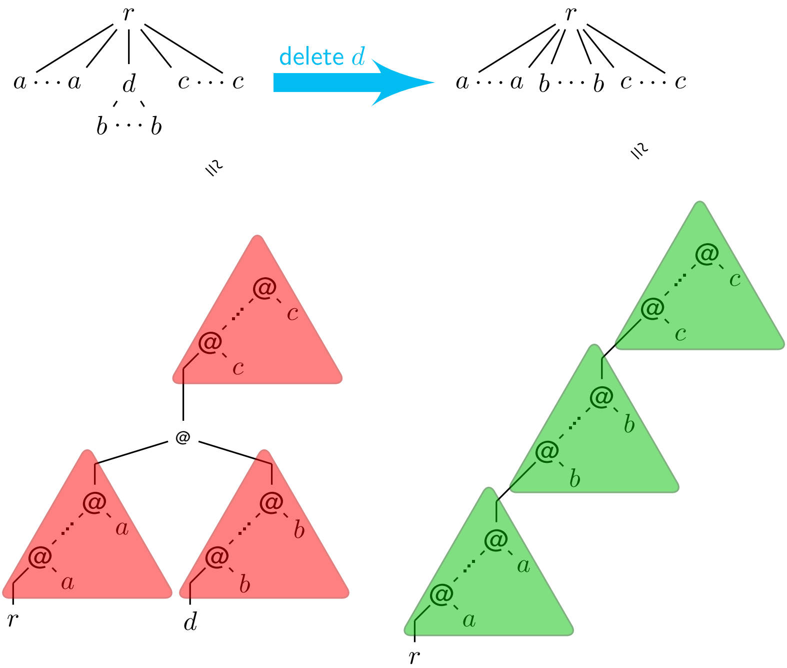
1. λ preserves contexts:
   contexts(X) $\subseteq$ contexts$(\lambda(X))$

2. λ respects post-order of components:
   $X \leq_{\text{postorder}} Y \ \leftrightarrow\ \lambda(X) \leq_{\text{postorder}} \lambda(Y)$

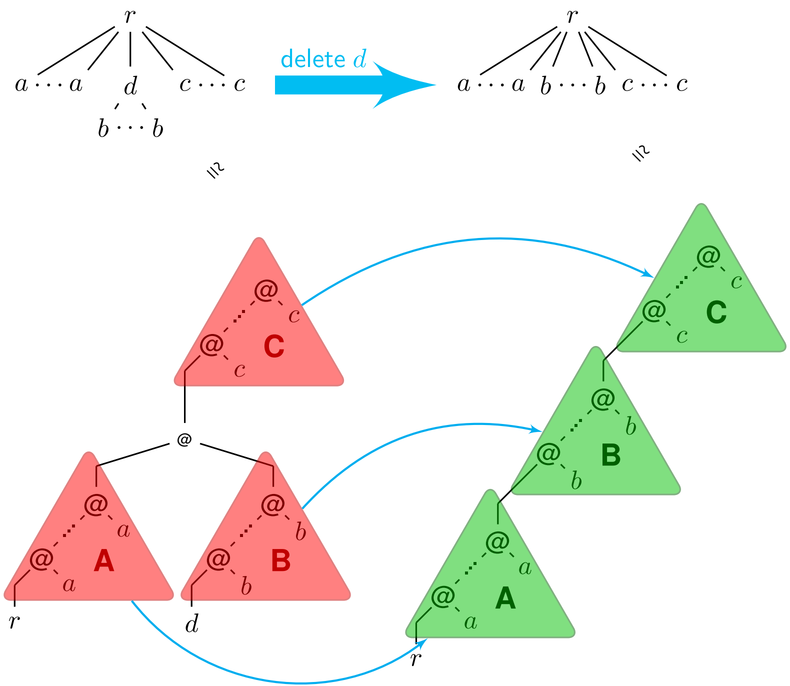
3. λ preserves ancestorship of vertical components:
   $X \leq_{\text{ancestor}} Y \ \leftrightarrow\ \lambda(X) \leq_{\text{ancestor}} \lambda(Y)$
   whenever vertical$-$contexts$(X) \neq \varnothing$

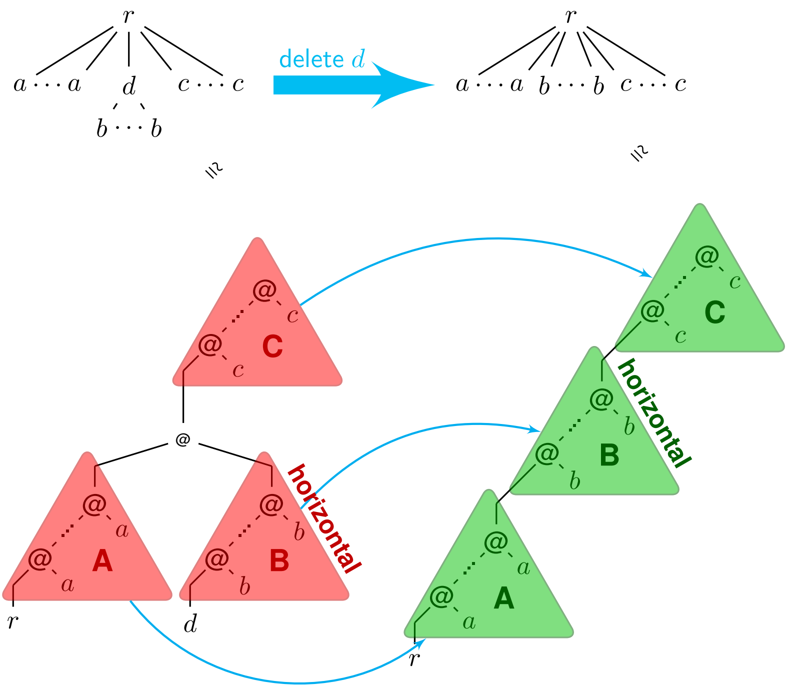

r
a ⋯ a
d
c ⋯ c
b ⋯ b
delete d
r
a ⋯ a
b ⋯ b
c ⋯ c


delete d



delete d


delete d
A
B
C
A
B
C



r
a ⋯ a
d
c ⋯ c
b ⋯ b
delete d
r
a ⋯ a
b ⋯ b
c ⋯ c
≅
≅
C
@
@
c
c
@
A
B
horizontal
r
d
C
B
A
horizontal
r

Complexity of **non-streaming** bounded repairability problem:

	det. DTD	DTD	stepwise
universal	**P**	**PSPACE**	**EXP**
fixed alphabet det. DTD	**coNP**	**PSPACE**	**PSPACE**
non recursive det. DTD	**coNEXP**	**coNEXP**	**coNEXP**
stepwise	**coNEXP**	**coNEXP**	**coNEXP**

Complexity of **non-streaming** bounded repairability problem:

	det. DTD	DTD	stepwise
universal	**P**	**PSPACE**	**EXP**
fixed alphabet det. DTD	**coNP**	**PSPACE**	**PSPACE**
non recursive det. DTD	**coNEXP**	**coNEXP**	**coNEXP**
stepwise	**coNEXP**	**coNEXP**	**coNEXP**

Complexity of **streaming** bounded repairability problem:

	det. DTD	DTD
universal	**P**	**PSPACE**
DTD	**EXP**	**EXP**

Some references...

- **Regular Repair of Specifications**
  Benedikt, Riveros, P. – LICS 2011
- **The cost of traveling between languages**
  Benedikt, Riveros, P. – ICALP 2011
- **Bounded repairability for regular tree languages**
  Riveros, Staworko, P. – ICDT 2012
- **Which DTDs are streaming bounded repairable?**
  Bourhis, Riveros, Staworko, P. – ICDT 2013

...and other related topics

- normalized edit cost $\sup_{s \in S} \min_{t \in T} \frac{\mathrm{dist}(s,t)}{|s|}$
- distance automata and limitedness problem
- energy games with perfect/imperfect information

Some references...

- **Regular Repair of Specifications**
  Benedikt, Riveros, P. – LICS 2011
- **The cost of traveling between languages**
  Benedikt, Riveros, P. – ICALP 2011
- **Bounded repairability for regular tree languages**
  Riveros, Staworko, P. – ICDT 2012
- **Which DTDs are streaming bounded repairable?**
  Bourhis, Riveros, Staworko, P. – ICDT 2013

...and other related topics

- normalized edit cost $\sup_{s \in S} \min_{t \in T} \frac{\mathrm{dist}(s,t)}{|s|}$
- distance automata and limitedness problem
- energy games with perfect/imperfect information