Deciding emptiness of TAGED

11 octobre 2010

Workshop WTS

Florent Jacquemard and C. Vacher

Inspired by Wiltord Charatonik

About this talk

- This talk is NOT about what the abstract in the programme mentions
- We present a subcase of the same problem
- We reach a primitive recursive (but still high) complexity
- We use another technique which is both new and old

Tree Automata with Global Equality and Disequality Consraints

Definition

A *Tree Automata with Global Equality and Disequality constraints* (TAGED) is a tuple $\langle Q, \Sigma, F, \Delta, \approx, \not\approx \rangle$ such that

- $\langle Q, \Sigma, F, \Delta \rangle$ is a TA
- $\approx$ is a reflexive symmetric relation on Q
- $\not\approx$ is an irreflexive and symmetric relation on Q

A TAGED is said *positive* (resp. *negative*) if $\approx$ (resp. $\not\approx$) is empty.

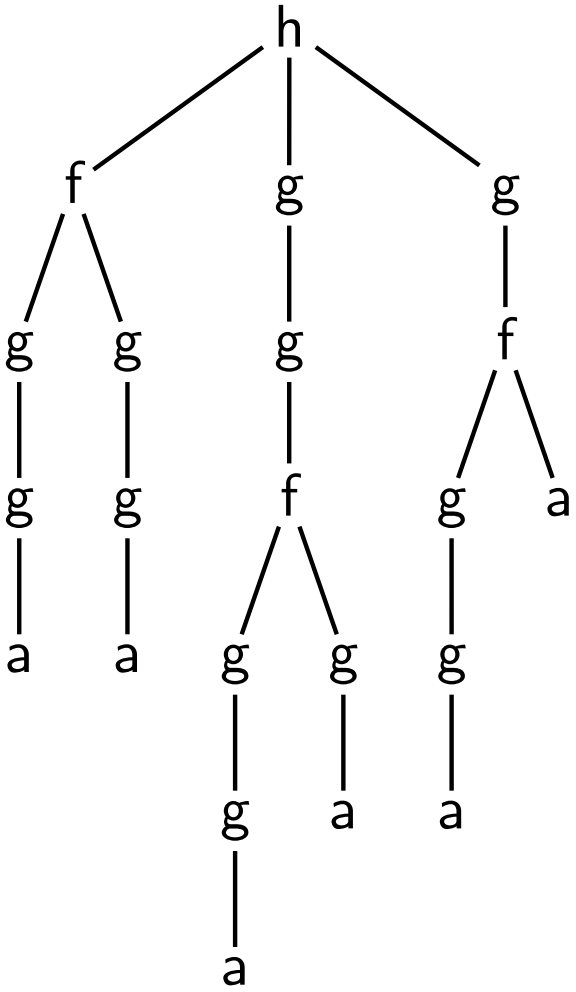
Accepting run of TAGED

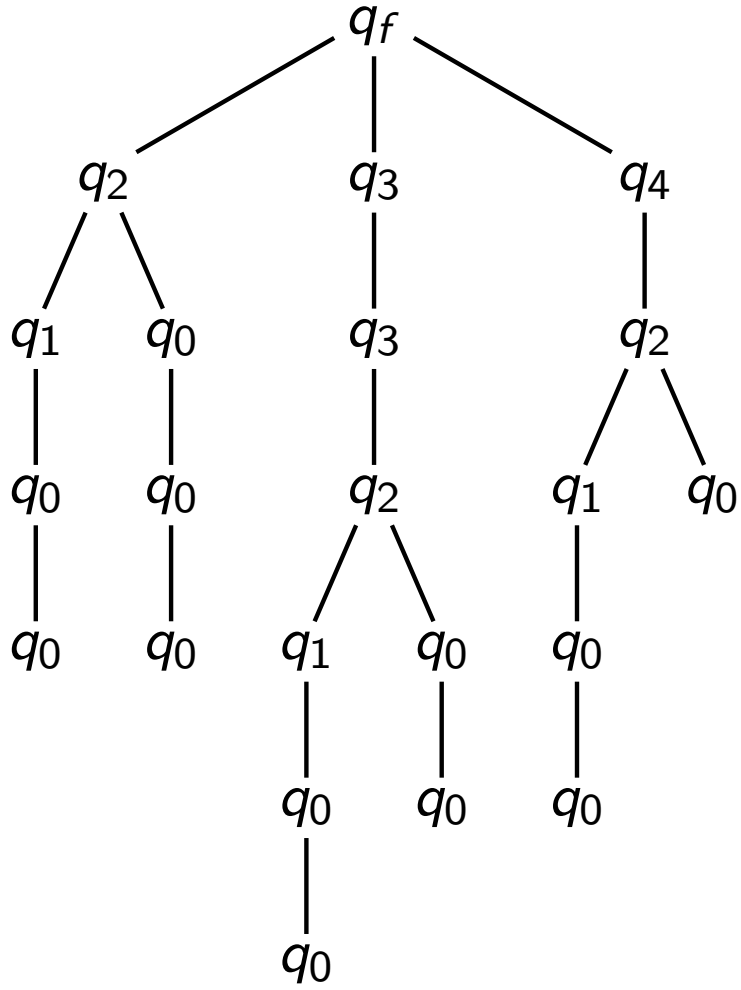
Definition

Let A be a TAGED and t a term. An *accepting run* of A on t is an accepting run r of the underlying TA such that

- for all states $q_1, q_2 \in Q, q_1 \approx q_2$, and for all positions p_1, p_2, $r[p_1] = q_1$ and $r[p_2] = q_2$ implies $t|_{p_1} = t|_{p_2}$
- for all states $q_1, q_2 \in Q, q_1 \not\approx q_2$, and for all positions p_1, p_2 $r[p_1] = q_1$ and $r[p_2] = q_2$ implies $t|_{p_1} \neq t|_{p_2}$

Example of a run





Note that $q_3 \not\approx q_4$, and $q_1 \approx q_1$

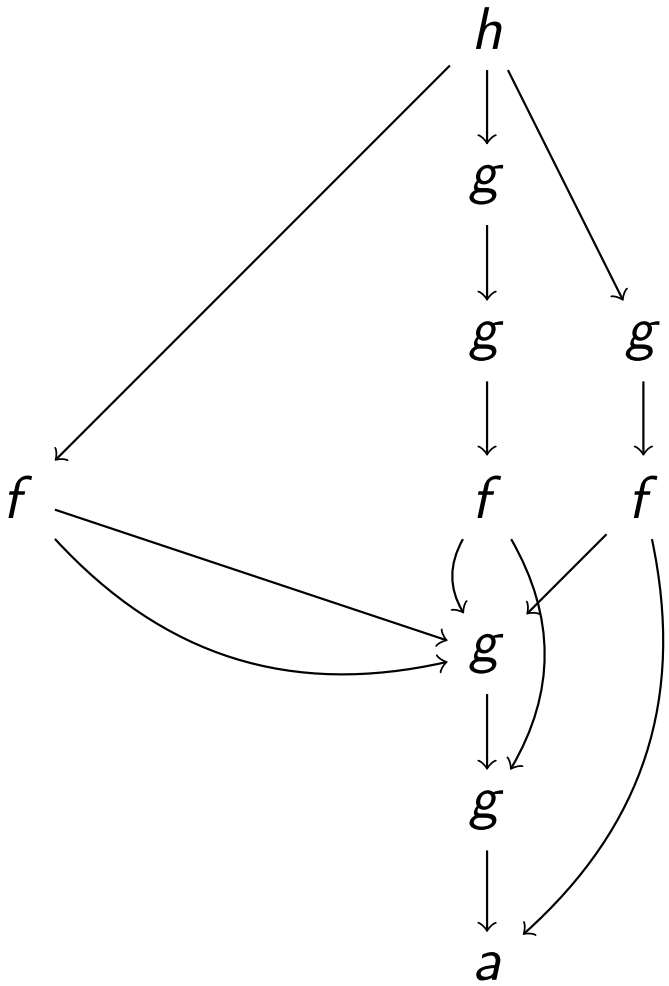
Known decision results(Filiot, Talbot, Tison 2008)

- Emptiness is EXPTIME-complete for positive TAGED
  Proof hint : reduction to a positive TAGED with only constraints of the form $q \approx q$. Emptiness for such an automaton is equivalent to emptiness for TA.
- Emptiness is NEXPTIME for negative TAGED
  Proof hint : reduction to a system of negative set constraints (Charatoniki, Pacholski 1993)
- Emptiness is 2NEXPTIME for vertically bounded TAGED (consider only run with a bounded number of disequality constraints in each branch)

DAG representation of terms

- Representation of a term as a directed ordered labeled acyclic graph $G = (V, succ, \lambda)$
- Respect the maximal sharing property (i.e. each subterm is represented by a single node, even if it occurs several times).
- Bijection between terms and directed ordered labeled acyclic graph with maximal sharing (t-dag for short)

Example of t-dag



t-dag automata

Definition

A *t-dag automaton* A is a tree automaton $\langle \Sigma, Q, F, \Delta \rangle$. A *run* of a t-dag automaton A on a t-dag $G = (V, succ, \lambda)$ is a mapping $r : V \to Q$ that respects the transition rules of Δ. It is successful if $r(u) \in F$ where u is the root of G.

- equivalent to a run on the term where each occurence of a term is labeled by the same state
- languages recognized by deterministic automata on terms and t-dags are equivalent

Negative TAGED $\equiv$ t-dag automata

Theorem

A language L of terms is recognizable by a negative TAGED if and only if the language L' of t-dags representing the terms of L is recognizable by a t-dag automaton.

Negative TAGED $\Leftarrow$ t-dag automata

- A t-dag automaton $A = \langle \Sigma, Q, F, \Delta \rangle$
- We define $q_1 \not\approx q_2 \equiv q_1 \neq q_2$
- Two different subterms are necessarily labeled by two different nodes

Negative TAGED $\Rightarrow$ t-dag automata

- A negative TAGED $A = \langle \Sigma, Q, F, \Delta, \not\approx \rangle$
- We define $Q' = 2^Q \setminus \{P \mid \exists q_1, q_2, q_1 \not\approx q_2, q_1, q_2 \in P\}$
- $F' = \{P \mid \exists q_f \in F, q_f \in P\}$
- $f(P_1, \ldots, P_n) \to P \in \Delta'$ if $\forall q \in P, \exists q_1 \in P_1, \ldots, q_n \in P_n$ such that $f(q_1, \ldots, q_n) \to q \in \Delta$
- a run of the t-dag automaton labels each nodes with all the states that label the occurences of the subterm in a run of the negative TAGED. No subterm is labeled by two nodes in relation of disequality

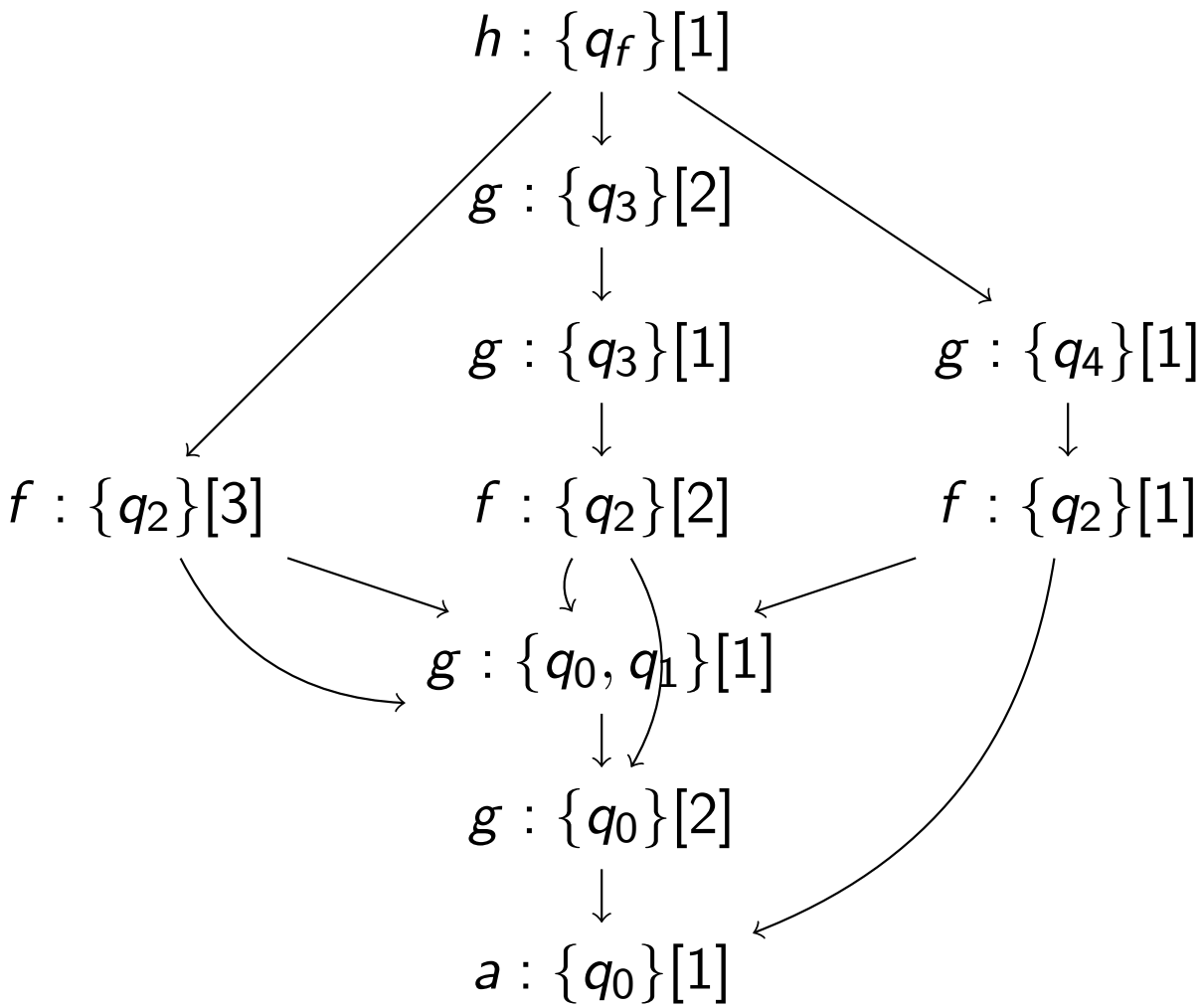
Emptiness of t-dag automata

- Emptiness is NEXPTIME-complete
- t-dag automata cannot encode TAGED (cannot force the use of the same term for a same node)
- Improvement of the proof in order to handle TAGED

The mapping $states_r$

- G a t-dag representing t, r a run of a TAGED A on t
- $states_r(v)$ the set of states labeling all occurences of the subterm represented by v

Example of $states_r$



Index and transition with pointers

- A linear ordering on subterms, compatible with the height ordering
- Give index to subterms : $P[1]$ is the smallest subterm whose node representation's mapping by $states_r$ is P
- extends this linear odering to tuples of nodes : $\sqsubset$
- Transitions with pointers $f(P_1[i_1], \ldots, P_n[i_n]) \ldots P$

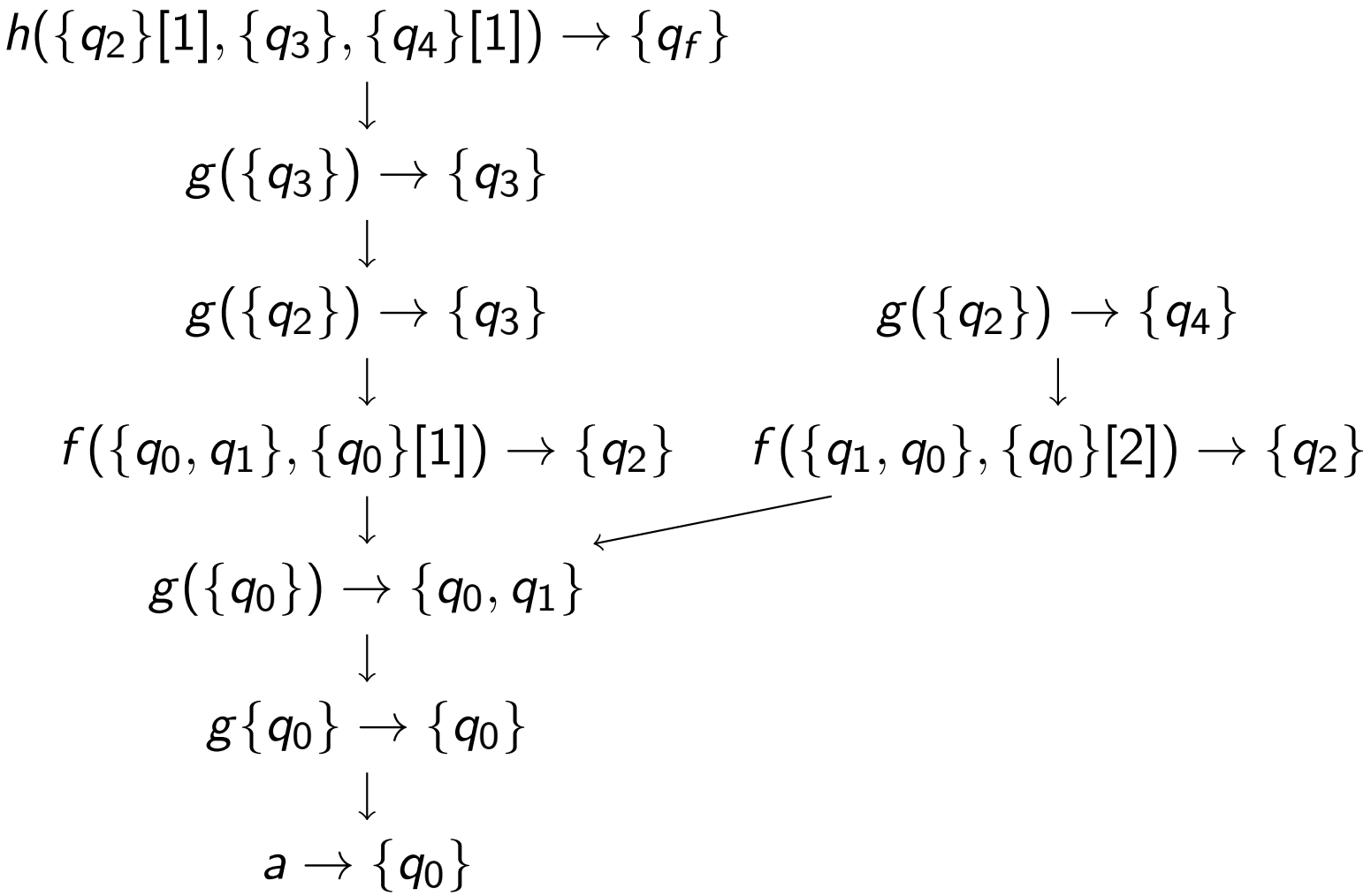
Fork of degree m

- we want to keep the longest path of a term
- use as few nodes as possibles for other childrens
- a node v with m predecessors $v_1, \ldots, v_m$
- $states_r(v_1) = \cdots = states_r(v_m)$
- v main successor of $v_1, \ldots, v_m$ at the same position k
- for all i, $states_r(succ(i, v_1)) = \cdots = states_r(succ(i, v_m)) = P_i$

Building a Skeleton

- Take the root of G and all its main path
- add nodes called *milestones* to handle forks.
- for a fork of degree m, we take the m lowest sequences of nodes labeled by the states $\langle P_1, \ldots, P_{k-1}, P_{k+1}, \ldots, P_n \rangle$
- we add the milestones and its main path to the skeleton

Example of skeleton



There are at most $2^{2|Q|}$ milestones

- Fork of degree $m \Rightarrow$ at most m milestones of index greater than !
- Degree of fork bounded by the width of the term at a given height
- Fork of degree m deletes $m-1$ paths and creates at most $m-1$ nodes of index > 1
- There are at most $2^{|Q|}$ milestones of index 1

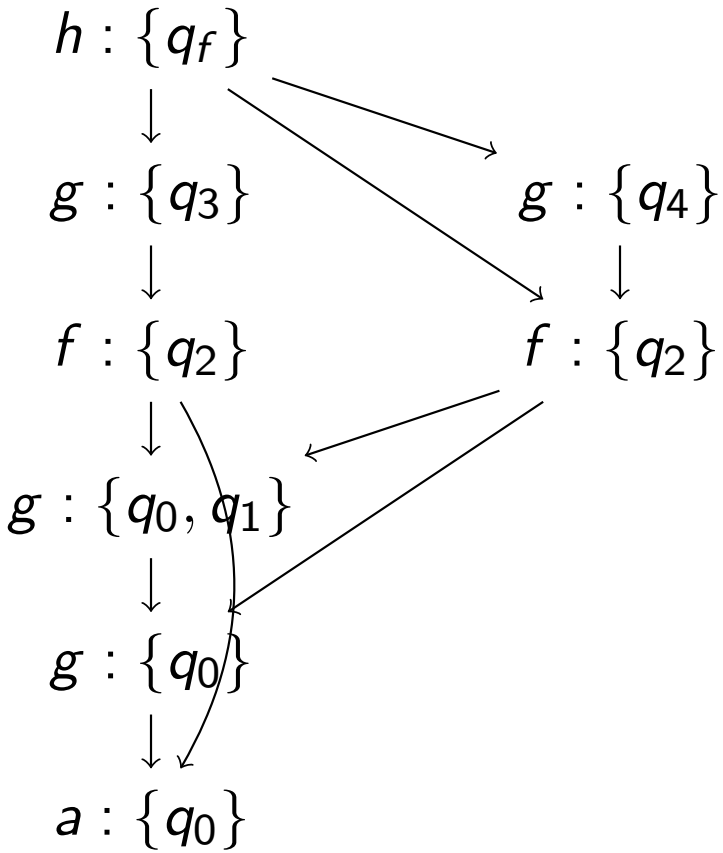
From skeleton to recognized t-dag

- consider edges instead of pointers
- the obtained graph is a t-dag (non-trivial)
- this t-dag represents a term recognized by the same TAGED (thanks to $states_r$)

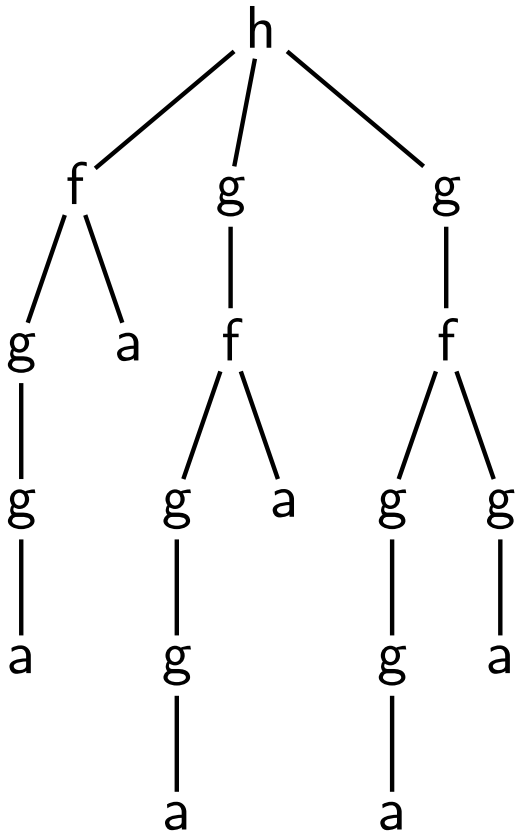
Having a small recognized t-dag

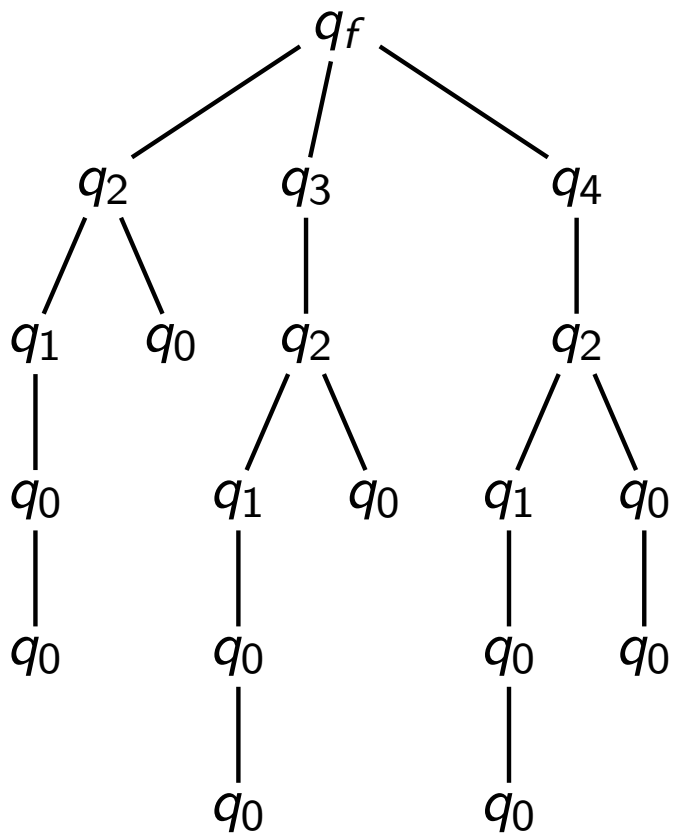
- bounded number of milestones $\not\Rightarrow$ bounded number of nodes
- we can pump on non-milestones nodes $\Rightarrow$ at most $2^{|Q|}$ nodes between two milestones
- t-dag with at most $2^{3|Q|}$ nodes representing a recognized term

Example of a pumping



The induced term and run





Conclusion

- We have a primitive recursive complexity for emptiness of TAGED
- We proved a strong link between automata with constraints and t-dag automata
- Do we have 2NEXPTIME-hardness ?