DAG Compressions

L. Gordeev

Uni-Tübingen, PUC Rio de Janeiro

Oberwolfach, November, 2011

§1. Term compression -1-

- *Dag-complexity* (or *graph-complexity*) of a given algebraic term $\mathfrak{t}$ is the minimal number of vertices in a rooted DAG (not necessarily a tree!) that represents $\mathfrak{t}$. This definition can be constructively specified, as follows.

- *Dag-complexity* (or *graph-complexity*) of a given algebraic term $\mathfrak{t}$ is the minimal number of vertices in a rooted DAG (not necessarily a tree!) that represents $\mathfrak{t}$. This definition can be constructively specified, as follows.

Definition

§1. Term compression -1-

- *Dag-complexity* (or *graph-complexity*) of a given algebraic term $\mathfrak{t}$ is the minimal number of vertices in a rooted DAG (not necessarily a tree!) that represents $\mathfrak{t}$. This definition can be constructively specified, as follows.

Definition

Consider arbitrary algebraic language $\mathcal{L}$ with individual variables, constants and (w.l.o.g.) binary function symbols. The *terms*:

- *Dag-complexity* (or *graph-complexity*) of a given algebraic term $\mathfrak{t}$ is the minimal number of vertices in a rooted DAG (not necessarily a tree!) that represents $\mathfrak{t}$. This definition can be constructively specified, as follows.

Definition

Consider arbitrary algebraic language $\mathcal{L}$ with individual variables, constants and (w.l.o.g.) binary function symbols. The *terms*:

1. Individual variables and constants are terms of the depth 0, also called *atoms*.

- *Dag-complexity* (or *graph-complexity*) of a given algebraic term $\mathfrak{t}$ is the minimal number of vertices in a rooted DAG (not necessarily a tree!) that represents $\mathfrak{t}$. This definition can be constructively specified, as follows.

Definition

Consider arbitrary algebraic language $\mathcal{L}$ with individual variables, constants and (w.l.o.g.) binary function symbols. The *terms*:

1. Individual variables and constants are terms of the depth 0, also called *atoms*.
2. If $\mathfrak{s}$, $\mathfrak{t}$ are any terms and f any function symbol, then $f(\mathfrak{s}, \mathfrak{t})$ is a term of the depth $1 + \max\{depth(\mathfrak{s}), depth(\mathfrak{t})\}$;

§1. Term compression -1-

- *Dag-complexity* (or *graph-complexity*) of a given algebraic term $\mathfrak{t}$ is the minimal number of vertices in a rooted DAG (not necessarily a tree!) that represents $\mathfrak{t}$. This definition can be constructively specified, as follows.

Definition

Consider arbitrary algebraic language $\mathcal{L}$ with individual variables, constants and (w.l.o.g.) binary function symbols. The *terms*:

1. Individual variables and constants are terms of the depth 0, also called *atoms*.
2. If $\mathfrak{s}$, $\mathfrak{t}$ are any terms and f any function symbol, then $f(\mathfrak{s}, \mathfrak{t})$ is a term of the depth $1 + \max\{depth(\mathfrak{s}), depth(\mathfrak{t})\}$; in the Łukasiewicz form we write $f\mathfrak{s}\mathfrak{t}$ instead of $f(\mathfrak{s}, \mathfrak{t})$.

Term compression -2-

Term compression -2-

Definition

For any term $\mathfrak{t}$ of $\mathcal{L}$ we define the *dag-complexity* of $\mathfrak{t}$, $\delta(\mathfrak{t})$.

Term compression -2-

Definition

For any term $\mathfrak{t}$ of $\mathcal{L}$ we define the *dag-complexity* of $\mathfrak{t}$, $\delta(\mathfrak{t})$.

1. $\delta(\mathfrak{t}) := \delta\{\mathfrak{t}\}$.

Term compression -2-

Definition

For any term $\mathfrak{t}$ of $\mathcal{L}$ we define the *dag-complexity* of $\mathfrak{t}$, $\delta(\mathfrak{t})$.

1. $\delta(\mathfrak{t}) := \delta\{\mathfrak{t}\}$.
2. $\delta\{\mathfrak{t}_1, \cdots, \mathfrak{t}_k\} := \#\{\mathfrak{t}_1, \cdots, \mathfrak{t}_k\}$, if $\mathfrak{t}_1$, ..., $\mathfrak{t}_k$ are atoms.

Term compression -2-

Definition

For any term $\mathfrak{t}$ of $\mathcal{L}$ we define the *dag-complexity* of $\mathfrak{t}$, $\delta(\mathfrak{t})$.

1. $\delta(\mathfrak{t}) := \delta\{\mathfrak{t}\}$.
2. $\delta\{\mathfrak{t}_1, \cdots, \mathfrak{t}_k\} := \#\{\mathfrak{t}_1, \cdots, \mathfrak{t}_k\}$, if $\mathfrak{t}_1$, ..., $\mathfrak{t}_k$ are atoms.
3. $\delta\{f(\mathfrak{s}, \mathfrak{t}), \mathfrak{t}_1, \cdots, \mathfrak{t}_k\} := 1 + \delta\{\mathfrak{s}, \mathfrak{t}, \mathfrak{t}_1, \cdots, \mathfrak{t}_k\}$, if the depth of $f(\mathfrak{s}, \mathfrak{t})$ is $\geq$ than maximal depth of $\mathfrak{t}_i$, $1 \leq i \leq k$.

Term compression -2-

Definition

For any term $\mathfrak{t}$ of $\mathcal{L}$ we define the *dag-complexity* of $\mathfrak{t}$, $\delta(\mathfrak{t})$.

1. $\delta(\mathfrak{t}) := \delta\{\mathfrak{t}\}$.
2. $\delta\{\mathfrak{t}_1, \cdots, \mathfrak{t}_k\} := \#\{\mathfrak{t}_1, \cdots, \mathfrak{t}_k\}$, if $\mathfrak{t}_1$, ..., $\mathfrak{t}_k$ are atoms.
3. $\delta\{f(\mathfrak{s}, \mathfrak{t}), \mathfrak{t}_1, \cdots, \mathfrak{t}_k\} := 1 + \delta\{\mathfrak{s}, \mathfrak{t}, \mathfrak{t}_1, \cdots, \mathfrak{t}_k\}$, if the depth of $f(\mathfrak{s}, \mathfrak{t})$ is $\geq$ than maximal depth of $\mathfrak{t}_i$, $1 \leq i \leq k$.

Term compression -2-

Definition

For any term $\mathfrak{t}$ of $\mathcal{L}$ we define the *dag-complexity* of $\mathfrak{t}$, $\delta(\mathfrak{t})$.

1. $\delta(\mathfrak{t}) := \delta\{\mathfrak{t}\}$.
2. $\delta\{\mathfrak{t}_1, \cdots, \mathfrak{t}_k\} := \#\{\mathfrak{t}_1, \cdots, \mathfrak{t}_k\}$, if $\mathfrak{t}_1$, ..., $\mathfrak{t}_k$ are atoms.
3. $\delta\{f(\mathfrak{s}, \mathfrak{t}), \mathfrak{t}_1, \cdots, \mathfrak{t}_k\} := 1 + \delta\{\mathfrak{s}, \mathfrak{t}, \mathfrak{t}_1, \cdots, \mathfrak{t}_k\}$, if the depth of $f(\mathfrak{s}, \mathfrak{t})$ is $\geq$ than maximal depth of $\mathfrak{t}_i$, $1 \leq i \leq k$.

- $\delta(\mathfrak{t})$ is easily computable e. g. in Maple.

Term compression -3-

Example

Consider term $\mathfrak{t} = f(g(x, f(y, h(x, y))), h(x, y))$ of the depth 4 in the language with variables x, y and function symbols f, g, h.

Term compression -3-

Example

Consider term $\mathfrak{t} = f(g(x, f(y, h(x, y))), h(x, y))$ of the depth 4 in the language with variables x, y and function symbols f, g, h.

We have

Term compression -3-

Example

Consider term $\mathfrak{t} = f(g(x, f(y, h(x, y))), h(x, y))$ of the depth 4 in the language with variables x, y and function symbols f, g, h.

We have

$$\begin{aligned}
\delta(\mathfrak{t}) &= \delta\{\mathfrak{t}\} = 1 + \delta\{g(x, f(y, h(x, y))), h(x, y)\} \\
&= 2 + \delta\{x, f(y, h(x, y)), h(x, y)\} \\
&= 3 + \delta\{x, y, h(x, y), h(x, y)\} \\
&= 3 + \delta\{x, y, h(x, y)\} = 4 + \delta\{x, y, x, y\} \\
&= 4 + \delta\{x, y\} = 4 + \#\{x, y\} \\
&= 4 + 2 = 6
\end{aligned}$$

Term compression -3-

Example

Consider term $\mathfrak{t} = f\left(g\left(x, f\left(y, h\left(x, y\right)\right)\right), h\left(x, y\right)\right)$ of the depth 4 in the language with variables x, y and function symbols f, g, h.

We have

$$\begin{aligned}
\delta\left(\mathfrak{t}\right) &= \delta\left\{\mathfrak{t}\right\} = 1 + \delta\left\{g\left(x, f\left(y, h\left(x, y\right)\right)\right), h\left(x, y\right)\right\} \\
&= 2 + \delta\left\{x, f\left(y, h\left(x, y\right)\right), h\left(x, y\right)\right\} \\
&= 3 + \delta\left\{x, y, h\left(x, y\right), h\left(x, y\right)\right\} \\
&= 3 + \delta\left\{x, y, h\left(x, y\right)\right\} = 4 + \delta\left\{x, y, x, y\right\} \\
&= 4 + \delta\left\{x, y\right\} = 4 + \#\left\{x, y\right\} \\
&= 4 + 2 = 6
\end{aligned}$$

Note that the ordinary Łukasiewicz length of $\mathfrak{t}$ is 11.

Term compression -4-

Term compression -4-

Example

Example

Define Fibonacci sequence of terms $\{F(i)\}_{i\geq 0}$ in the language $\mathcal{L}_{\mathrm{F}}$ with two individual constants 0, 1 and one function symbol $+$;

Example

Define Fibonacci sequence of terms $\{F(i)\}_{i\geq 0}$ in the language $\mathcal{L}_{\mathrm{F}}$ with two individual constants 0, 1 and one function symbol $+$; (use standard infix notation $\mathfrak{s}+\mathfrak{t}$ instead of Łukasiewicz $+\mathfrak{s}\mathfrak{t}$).

Example

Define Fibonacci sequence of terms $\{F(i)\}_{i\geq 0}$ in the language $\mathcal{L}_{\mathrm{F}}$ with two individual constants 0, 1 and one function symbol $+$; (use standard infix notation $\mathfrak{s}+\mathfrak{t}$ instead of Łukasiewicz $+\mathfrak{s}\mathfrak{t}$).

Example

Define Fibonacci sequence of terms $\{F(i)\}_{i\geq 0}$ in the language $\mathcal{L}_{\mathrm{F}}$ with two individual constants 0, 1 and one function symbol $+$; (use standard infix notation $\mathfrak{s}+\mathfrak{t}$ instead of Łukasiewicz $+\mathfrak{st}$).

- $F(0) := 0, F(1) := 1, F(i+2) := F(i) + F(i+1)$

Term compression -4-

Example

Define Fibonacci sequence of terms $\{F(i)\}_{i \geq 0}$ in the language $\mathcal{L}_{\mathrm{F}}$ with two individual constants 0, 1 and one function symbol $+$; (use standard infix notation $\mathfrak{s} + \mathfrak{t}$ instead of Łukasiewicz $+\mathfrak{st}$).

- $F(0) := 0, F(1) := 1, F(i+2) := F(i) + F(i+1)$
- The ordinary length of $F(i)$ slightly exceeds the i^{th} Fibonacci number, thus being exponential in i.

Term compression -4-

Example

Define Fibonacci sequence of terms $\{F(i)\}_{i\geq 0}$ in the language $\mathcal{L}_{\mathrm{F}}$ with two individual constants 0, 1 and one function symbol $+$; (use standard infix notation $\mathfrak{s}+\mathfrak{t}$ instead of Łukasiewicz $+\mathfrak{s}\mathfrak{t}$).

- $F(0) := 0, F(1) := 1, F(i+2) := F(i) + F(i+1)$
- The ordinary length of $F(i)$ slightly exceeds the i^{th} Fibonacci number, thus being exponential in i.
- But the corresponding dag-complexity is merely linear in i :

Term compression -4-

Example

Define Fibonacci sequence of terms $\{F(i)\}_{i\geq 0}$ in the language $\mathcal{L}_{\mathrm{F}}$ with two individual constants 0, 1 and one function symbol $+$; (use standard infix notation $\mathfrak{s}+\mathfrak{t}$ instead of Łukasiewicz $+\mathfrak{s}\mathfrak{t}$).

- $F(0) := 0, F(1) := 1, F(i+2) := F(i) + F(i+1)$
- The ordinary length of $F(i)$ slightly exceeds the i^{th} Fibonacci number, thus being exponential in i.
- But the corresponding dag-complexity is merely linear in i :

  $\delta(F(0)) = \delta(F(1)) = 1$ and $\delta(F(i)) = i+1$ for all $i > 1$.

§2. Generalizatons

§2. Generalizatons

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y} :=$ the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y}$:= the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

1. $D \rhd_0 D'$:

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y}$:= the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

1. $D \rhd_0 D'$: D' arises from D by identifying all leaves having the same labels and all vertices x, y such that $\ell(x) = \ell(y)$ and $(x \to y) \in D$. If not applicable, let $D' := D$.

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y}$:= the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

1. $D \rhd_0 D'$: D' arises from D by identifying all leaves having the same labels and all vertices x, y such that $\ell(x) = \ell(y)$ and $(x \to y) \in D$. If not applicable, let $D' := D$.
2. $D \rhd_1 D'$ (*contraction*),

§2. Generalizatons

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y}$:= the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

1. $D \rhd_0 D'$: D' arises from D by identifying all leaves having the same labels and all vertices x, y such that $\ell(x) = \ell(y)$ and $(x \to y) \in D$. If not applicable, let $D' := D$.
2. $D \rhd_1 D'$ (*contraction*), where ℓ is the labeling function:

§2. Generalizatons

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y}$:= the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

1. $D \rhd_0 D'$: D' arises from D by identifying all leaves having the same labels and all vertices x, y such that $\ell(x) = \ell(y)$ and $(x \rightarrow y) \in D$. If not applicable, let $D' := D$.
2. $D \rhd_1 D'$ (*contraction*), where ℓ is the labeling function:
    - Let x, y be vertices in D closest to the root such that: $\ell(x)\, R \ell(y)$, $(x \rightarrow y) \notin D$, $D_{>y}$ is a tree and $x \in D_{>y}$.

§2. Generalizatons

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y}$:= the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

1. $D \rhd_0 D'$: D' arises from D by identifying all leaves having the same labels and all vertices x, y such that $\ell(x) = \ell(y)$ and $(x \to y) \in D$. If not applicable, let $D' := D$.
2. $D \rhd_1 D'$ (*contraction*), where ℓ is the labeling function:
   - Let x, y be vertices in D closest to the root such that: $\ell(x) R \ell(y)$, $(x \to y) \notin D$, $D_{>y}$ is a tree and $x \in D_{>y}$.
   - D' := replace in D whole path $x \rightsquigarrow y$ by new edge $x \to y$.

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y}$:= the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

1. $D \rhd_0 D'$: D' arises from D by identifying all leaves having the same labels and all vertices x, y such that $\ell(x) = \ell(y)$ and $(x \to y) \in D$. If not applicable, let $D' := D$.
2. $D \rhd_1 D'$ (*contraction*), where ℓ is the labeling function:
   - Let x, y be vertices in D closest to the root such that: $\ell(x) R \ell(y)$, $(x \to y) \notin D$, $D_{>y}$ is a tree and $x \in D_{>y}$.
   - D' := replace in D whole path $x \rightsquigarrow y$ by new edge $x \to y$.
3. $D \rhd_2 D'$ (*conglutination*):

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y}$:= the sub-dag of $z \neq y$ having a path $z \leadsto y$.

1. $D \rhd_0 D'$: D' arises from D by identifying all leaves having the same labels and all vertices x, y such that $\ell(x) = \ell(y)$ and $(x \rightarrow y) \in D$. If not applicable, let $D' := D$.
2. $D \rhd_1 D'$ (*contraction*), where ℓ is the labeling function:
    - Let x, y be vertices in D closest to the root such that: $\ell(x) R \ell(y)$, $(x \rightarrow y) \notin D$, $D_{>y}$ is a tree and $x \in D_{>y}$.
    - $D' :=$ replace in D whole path $x \leadsto y$ by new edge $x \rightarrow y$.
3. $D \rhd_2 D'$ (*conglutination*):
    - Let x, y be vertices in D closest to the root such that: $\ell(x) R \ell(y)$, $D_{>y} \neq \emptyset$ is a tree and $x \notin D_{>y}$.

- Define reductions $\rhd_0, \rhd_1, \rhd_2$ on finite labeled rooted dag's D with reflexive and transitive binary relation R on labels, where $D_{>y} :=$ the sub-dag of $z \neq y$ having a path $z \rightsquigarrow y$.

1. $D \rhd_0 D'$: D' arises from D by identifying all leaves having the same labels and all vertices x, y such that $\ell(x) = \ell(y)$ and $(x \rightarrow y) \in D$. If not applicable, let $D' := D$.
2. $D \rhd_1 D'$ (*contraction*), where ℓ is the labeling function:
   - Let x, y be vertices in D closest to the root such that: $\ell(x) R \ell(y)$, $(x \rightarrow y) \notin D$, $D_{>y}$ is a tree and $x \in D_{>y}$.
   - $D' :=$ replace in D whole path $x \rightsquigarrow y$ by new edge $x \rightarrow y$.
3. $D \rhd_2 D'$ (*conglutination*):
   - Let x, y be vertices in D closest to the root such that: $\ell(x) R \ell(y)$, $D_{>y} \neq \emptyset$ is a tree and $x \notin D_{>y}$.
   - $D' := D$ plus new edge $x \rightarrow y$ minus $z \in D_{>y}$.

Dag-compressions, dag-complexity

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearlyly every $\rhd_i$ is size-reducing:

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearlyly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearlyly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

1. $\rhd_i$-irreducible dag's are called *normal*.

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

1. $\rhd_i$-irreducible dag's are called *normal*.
2. Let $\mathcal{T}$ be set of finite labeled rooted trees, R reflexive and transitive binary relation on labels.

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

1. $\rhd_i$-irreducible dag's are called *normal*.
2. Let $\mathcal{T}$ be set of finite labeled rooted trees, R reflexive and transitive binary relation on labels. For every tree $T \in \mathcal{T}$ there are chains of dag's $T = D_0 \rhd_{i_1} \cdots \rhd_{i_k} D_k \rhd_0 D_{k+1}$ $(k \geq 0, i_j = 1, 2)$ with normal D_{k+1}.

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearlyly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

1. $\rhd_i$-irreducible dag's are called *normal*.
2. Let $\mathcal{T}$ be set of finite labeled rooted trees, R reflexive and transitive binary relation on labels. For every tree $T \in \mathcal{T}$ there are chains of dag's $T = D_0 \rhd_{i_1} \cdots \rhd_{i_k} D_k \rhd_0 D_{k+1}$ $(k \geq 0, i_j = 1, 2)$ with normal D_{k+1}. Clearly $\#D_{k+1} \leq \#T$. Call these D_{k+1} *normal dag-like compressions* of T.

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearlyly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

1. $\rhd_i$-irreducible dag's are called *normal*.
2. Let $\mathcal{T}$ be set of finite labeled rooted trees, R reflexive and transitive binary relation on labels. For every tree $T \in \mathcal{T}$ there are chains of dag's $T = D_0 \rhd_{i_1} \cdots \rhd_{i_k} D_k \rhd_0 D_{k+1}$ $(k \geq 0, i_j = 1, 2)$ with normal D_{k+1}. Clearly $\#D_{k+1} \leq \#T$. Call these D_{k+1} *normal dag-like compressions* of T.
3. Let $\delta(T) := \min(\#D)$ for D ranging over normal dag-like compressions of T.

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearlyly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

1. $\rhd_i$-irreducible dag's are called *normal*.
2. Let $\mathcal{T}$ be set of finite labeled rooted trees, R reflexive and transitive binary relation on labels. For every tree $T \in \mathcal{T}$ there are chains of dag's $T = D_0 \rhd_{i_1} \cdots \rhd_{i_k} D_k \rhd_0 D_{k+1}$ $(k \geq 0, i_j = 1, 2)$ with normal D_{k+1}. Clearly $\#D_{k+1} \leq \#T$. Call these D_{k+1} *normal dag-like compressions* of T.
3. Let $\delta(T) := \min(\#D)$ for D ranging over normal dag-like compressions of T. Call $\delta(T)$ the *dag-complexity* of T.

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearlyly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

1. $\rhd_i$-irreducible dag's are called *normal*.
2. Let $\mathcal{T}$ be set of finite labeled rooted trees, R reflexive and transitive binary relation on labels. For every tree $T \in \mathcal{T}$ there are chains of dag's $T = D_0 \rhd_{i_1} \cdots \rhd_{i_k} D_k \rhd_0 D_{k+1}$ $(k \geq 0, i_j = 1, 2)$ with normal D_{k+1}. Clearly $\#D_{k+1} \leq \#T$. Call these D_{k+1} *normal dag-like compressions* of T.
3. Let $\delta(T) := \min(\#D)$ for D ranging over normal dag-like compressions of T. Call $\delta(T)$ the *dag-complexity* of T.
4. For any label Γ, let $\delta(\Gamma) := \min(\delta(T))$ for T ranging over $T \in \mathcal{T}$ with root-label Γ.

Dag-compressions, dag-complexity

Let $\#D$ be standard size of D. Clearlyly every $\rhd_i$ is size-reducing:

- $D \rhd_i D' \Rightarrow \#D > \#D'$, except $i = 0$ and $D = D'$.

Definition

1. $\rhd_i$-irreducible dag's are called *normal*.
2. Let $\mathcal{T}$ be set of finite labeled rooted trees, R reflexive and transitive binary relation on labels. For every tree $T \in \mathcal{T}$ there are chains of dag's $T = D_0 \rhd_{i_1} \cdots \rhd_{i_k} D_k \rhd_0 D_{k+1}$ $(k \geq 0, i_j = 1, 2)$ with normal D_{k+1}. Clearly $\#D_{k+1} \leq \#T$. Call these D_{k+1} *normal dag-like compressions* of T.
3. Let $\delta(T) := \min(\#D)$ for D ranging over normal dag-like compressions of T. Call $\delta(T)$ the *dag-complexity* of T.
4. For any label Γ, let $\delta(\Gamma) := \min(\delta(T))$ for T ranging over $T \in \mathcal{T}$ with root-label Γ. Call $\delta(\Gamma)$ the *dag-complexity* of Γ.

Dag-complexity problems

Dag-complexity problems

Problem

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

Problem

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

Problem

2. How to estimate $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

Problem

2. How to estimate $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

- **Term algebra** ($\delta(T) \cong \delta(\Gamma)$, see Chapter 1)

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

Problem

2. How to estimate $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

- **Term algebra** ($\delta(T) \cong \delta(\Gamma)$, see Chapter 1)
  - **Problem 1**: Easy (see Chapter 1).

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

Problem

2. How to estimate $\delta(T)$ *and/or* $\delta(\Gamma)$ *?*

- **Term algebra** ($\delta(T) \cong \delta(\Gamma)$, see Chapter 1)
  - **Problem 1**: Easy (see Chapter 1).
  - **Problem 2**: Roughly $\#T \geq \delta(T) \geq \log \#T$.

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ and/or $\delta(\Gamma)$?

Problem

2. How to estimate $\delta(T)$ and/or $\delta(\Gamma)$?

- **Term algebra** ($\delta(T) \cong \delta(\Gamma)$, see Chapter 1)
  - **Problem 1**: Easy (see Chapter 1).
  - **Problem 2**: Roughly $\#T \geq \delta(T) \geq \log \#T$.
    In most interesting cases $\#T$ exponential in $\delta(T)$
    (see Chapter 1).

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ and/or $\delta(\Gamma)$?

Problem

2. How to estimate $\delta(T)$ and/or $\delta(\Gamma)$?

- **Term algebra** ($\delta(T) \cong \delta(\Gamma)$, see Chapter 1)
  - **Problem 1**: Easy (see Chapter 1).
  - **Problem 2**: Roughly $\#T \geq \delta(T) \geq \log \#T$.
    In most interesting cases $\#T$ exponential in $\delta(T)$
    (see Chapter 1).
- **Generalizations**: Both problems are hard.

Dag-complexity problems

Problem

1. How to compute $\delta(T)$ and/or $\delta(\Gamma)$?

Problem

2. How to estimate $\delta(T)$ and/or $\delta(\Gamma)$?

- **Term algebra** ($\delta(T) \cong \delta(\Gamma)$, see Chapter 1)
  - **Problem 1**: Easy (see Chapter 1).
  - **Problem 2**: Roughly $\#T \geq \delta(T) \geq \log \#T$.
    In most interesting cases $\#T$ exponential in $\delta(T)$ (see Chapter 1).
- **Generalizations**: Both problems are hard.
  Proof theory provides most interesting applications.

Proof-theoretic interpretation

Proof-theoretic interpretation

- Let $\mathcal{S}$ be given finite collection of axioms and inference rules, as usual in proof theory.

Proof-theoretic interpretation

- Let $\mathcal{S}$ be given finite collection of axioms and inference rules, as usual in proof theory.
- Let $\mathcal{T}$ and $\mathcal{D}$ contain resp. *tree-like* and *dag-like proofs* (or *deductions*) as $\mathcal{S}$-generated trees, resp. dag's, labeled by, say, *sequents* (Γ, Γ', etc.).

Proof-theoretic interpretation

- Let $\mathcal{S}$ be given finite collection of axioms and inference rules, as usual in proof theory.
- Let $\mathcal{T}$ and $\mathcal{D}$ contain resp. *tree-like* and *dag-like proofs* (or *deductions*) as $\mathcal{S}$-generated trees, resp. dag's, labeled by, say, *sequents* (Γ, Γ', etc.).
- Let $\Gamma R \Gamma' :\Leftrightarrow \Gamma' = \theta(\Gamma)$ for $\theta \in Hom(Seq \rightarrow Seq)$,

Proof-theoretic interpretation

- Let $\mathcal{S}$ be given finite collection of axioms and inference rules, as usual in proof theory.
- Let $\mathcal{T}$ and $\mathcal{D}$ contain resp. *tree-like* and *dag-like proofs* (or *deductions*) as $\mathcal{S}$-generated trees, resp. dag's, labeled by, say, *sequents* (Γ, Γ', etc.).
- Let $\Gamma R \Gamma' :\Leftrightarrow \Gamma' = \theta(\Gamma)$ for $\theta \in Hom(Seq \rightarrow Seq)$, provided that sequent-homomorpism θ preserves provability.

Proof-theoretic interpretation

- Let $\mathcal{S}$ be given finite collection of axioms and inference rules, as usual in proof theory.
- Let $\mathcal{T}$ and $\mathcal{D}$ contain resp. *tree-like* and *dag-like proofs* (or *deductions*) as $\mathcal{S}$-generated trees, resp. dag's, labeled by, say, *sequents* (Γ, Γ', etc.).
- Let $\Gamma R \Gamma' :\Leftrightarrow \Gamma' = \theta(\Gamma)$ for $\theta \in Hom(Seq \to Seq)$, provided that sequent-homomorpism θ preserves provability.
- Define as above *normal dag-like compressions* $D \in \mathcal{D}$ of $T \in \mathcal{T}$ obtained by the chains of $\rhd_i$-reductions w.r.t. R.

Proof-theoretic interpretation

- Let $\mathcal{S}$ be given finite collection of axioms and inference rules, as usual in proof theory.
- Let $\mathcal{T}$ and $\mathcal{D}$ contain resp. *tree-like* and *dag-like proofs* (or *deductions*) as $\mathcal{S}$-generated trees, resp. dag's, labeled by, say, *sequents* (Γ, Γ', etc.).
- Let $\Gamma R \Gamma' :\Leftrightarrow \Gamma' = \theta(\Gamma)$ for $\theta \in Hom(Seq \rightarrow Seq)$, provided that sequent-homomorpism θ preserves provability.
- Define as above *normal dag-like compressions* $D \in \mathcal{D}$ of $T \in \mathcal{T}$ obtained by the chains of $\rhd_i$-reductions w.r.t. R.
- These normal dag-like compressions are the desired smallest dag-like deductions, while

Proof-theoretic interpretation

- Let $\mathcal{S}$ be given finite collection of axioms and inference rules, as usual in proof theory.
- Let $\mathcal{T}$ and $\mathcal{D}$ contain resp. *tree-like* and *dag-like proofs* (or *deductions*) as $\mathcal{S}$-generated trees, resp. dag's, labeled by, say, *sequents* (Γ, Γ', etc.).
- Let $\Gamma R \Gamma' :\Leftrightarrow \Gamma' = \theta(\Gamma)$ for $\theta \in Hom(Seq \rightarrow Seq)$, provided that sequent-homomorpism θ preserves provability.
- Define as above *normal dag-like compressions* $D \in \mathcal{D}$ of $T \in \mathcal{T}$ obtained by the chains of $\rhd_i$-reductions w.r.t. R.
- These normal dag-like compressions are the desired smallest dag-like deductions, while
- $\delta(T)$ is "true" dag-complexity of given tree-like deduction T.

Proof search connections

Proof search connections

Most interesting case: cutfree proof systems.

Proof search connections

Most interesting case: cutfree proof systems.

- Let $\mathcal{T}$ contain *cutfree* tree-like deductions.

Proof search connections

Most interesting case: cutfree proof systems.

- Let $\mathcal{T}$ contain *cutfree* tree-like deductions.
  - By Gentzen-style cut elimination results this is not really a restriction (in pure logic, at least).

Proof search connections

Most interesting case: cutfree proof systems.

- Let $\mathcal{T}$ contain *cutfree* tree-like deductions.
  - By Gentzen-style cut elimination results this is not really a restriction (in pure logic, at least).
  - However there are significant proof complexity implications (re: "speed-up", to be discussed later).

Proof search connections

Most interesting case: cutfree proof systems.

- Let $\mathcal{T}$ contain *cutfree* tree-like deductions.
  - By Gentzen-style cut elimination results this is not really a restriction (in pure logic, at least).
  - However there are significant proof complexity implications (re: "speed-up", to be discussed later).
- Important advantage: cutfree tree-like proof systems admit reasonable (semi-)automated semi-analytic proof search (re: Gentzen-style *subformula property*).

Proof search connections

Most interesting case: cutfree proof systems.

- Let $\mathcal{T}$ contain *cutfree* tree-like deductions.
  - By Gentzen-style cut elimination results this is not really a restriction (in pure logic, at least).
  - However there are significant proof complexity implications (re: "speed-up", to be discussed later).
- Important advantage: cutfree tree-like proof systems admit reasonable (semi-)automated semi-analytic proof search (re: Gentzen-style *subformula property*).
- Our dag-like compressions D preserve this advantage, provided that R is sufficiently constructive.

Proof search connections

Most interesting case: cutfree proof systems.

- Let $\mathcal{T}$ contain *cutfree* tree-like deductions.
  - By Gentzen-style cut elimination results this is not really a restriction (in pure logic, at least).
  - However there are significant proof complexity implications (re: "speed-up", to be discussed later).
- Important advantage: cutfree tree-like proof systems admit reasonable (semi-)automated semi-analytic proof search (re: Gentzen-style *subformula property*).
- Our dag-like compressions D preserve this advantage, provided that R is sufficiently constructive.
  - However D may depend on the choice of $\rhd_2$ involved; thus the sources T can have different normal forms.

Propositional logic

Further specifications:

Propositional logic

Further specifications:

1. Classical propositional logic.

Propositional logic

Further specifications:

1. Classical propositional logic.
2. DNF logic.

Propositional logic

Further specifications:

1. Classical propositional logic.
2. DNF logic.
3. Relation R as homomorphism generated by *variables* $\rightarrow$ *literals* substitutions and suitable weakenings.

Propositional logic

Further specifications:

1. Classical propositional logic.
2. DNF logic.
3. Relation R as homomorphism generated by *variables* $\rightarrow$ *literals* substitutions and suitable weakenings.

- (1) is polynomially reducible to (2), so consider (2) & (3).

Propositional logic

Further specifications:

1. Classical propositional logic.
2. DNF logic.
3. Relation R as homomorphism generated by *variables* $\longrightarrow$ *literals* substitutions and suitable weakenings.

- (1) is polynomially reducible to (2), so consider (2) & (3).
- **Example**: Very efficient sequent calculus for DNF tautologies, called $\mathrm{SEQ}_{\mathrm{TAU}}$.

§3. Example: SEQ_{TAU}

§3. Example: $\mathrm{SEQ}_{\mathrm{TAU}}$

- Sequents: $\Gamma = M_1, \cdots, M_s$ where $M_i \subset_{\mathrm{fin}} \mathbb{Z}_0 := \mathbb{Z} - \{0\}$ such that $(\forall a, b \in M_i)\,(a + b \neq 0)$

§3. Example: $\mathrm{SEQ}_{\mathrm{TAU}}$

- Sequents: $\Gamma = M_1, \cdots, M_s$ where $M_i \subset_{\text{fin}} \mathbb{Z}_0 := \mathbb{Z} - \{0\}$ such that $(\forall a, b \in M_i)\,(a + b \neq 0)$
- *Axiom* $(\mathrm{A}_1): \ \{1\}, \{-1\}$

§3. Example: $\mathrm{SEQ}_{\mathrm{TAU}}$

- Sequents: $\Gamma = M_1, \cdots, M_s$ where $M_i \subset_{\mathrm{fin}} \mathbb{Z}_0 := \mathbb{Z} - \{0\}$ such that $(\forall a, b \in M_i)\,(a + b \neq 0)$
- *Axiom* $(\mathrm{A}_1):\ \{1\}, \{-1\}$
- *Weakening rules*

$$(\mathrm{W}_1):\ \frac{\Gamma}{M, \Gamma} \qquad , \qquad (\mathrm{W}_2):\ \frac{M \cup M', \Gamma}{M, \Gamma}$$

§3. Example: $\mathrm{SEQ}_{\mathrm{TAU}}$

- Sequents: $\Gamma = M_1, \cdots, M_s$ where $M_i \subset_{\text{fin}} \mathbb{Z}_0 := \mathbb{Z} - \{0\}$ such that $(\forall a, b \in M_i)\,(a + b \neq 0)$
- *Axiom* $(\mathrm{A}_1) : \ \{1\}, \{-1\}$
- *Weakening rules*

$$(\mathrm{W}_1) : \ \frac{\Gamma}{M, \Gamma} \qquad , \qquad (\mathrm{W}_2) : \ \frac{M \cup M', \Gamma}{M, \Gamma}$$

- *Substitution rule*

$$(\mathrm{S}) : \ \frac{\Gamma}{\theta(\Gamma)}, \ \textit{where } \theta \in \textit{Hom}\,(\mathrm{Seq} \to \mathrm{Seq})$$

§3. Example: $\mathrm{SEQ}_{\mathrm{TAU}}$

- Sequents: $\Gamma = M_1, \cdots, M_s$ where $M_i \subset_{\text{fin}} \mathbb{Z}_0 := \mathbb{Z} - \{0\}$ such that $(\forall a, b \in M_i)\,(a + b \neq 0)$
- *Axiom* $(\ \mathrm{A}_1) : \ \{1\}, \{-1\}$
- *Weakening rules*

$$(\mathrm{W}_1) : \ \frac{\Gamma}{M, \Gamma} \qquad , \qquad (\mathrm{W}_2) : \ \frac{M \cup M', \Gamma}{M, \Gamma}$$

- *Substitution rule*

$$(\mathrm{S}) : \ \frac{\Gamma}{\theta\,(\Gamma)}, \ \textit{where } \theta \in \textit{Hom}\,(\mathrm{Seq} \to \mathrm{Seq})$$

- *Main rule*, where $\pm k \notin M_i, M'_j, \Gamma$

$$(\mathrm{Q}) : \ \frac{M_1, \cdots, M_r, \Gamma \qquad\qquad M'_1, \cdots, M'_{r'}, \Gamma}{\{k\} \cup M_1, \cdots, \{k\} \cup M_r, \{-k\} \cup M'_1, \cdots, \{-k\} \cup M'_{r'}, \Gamma}$$

§3. Example: $\mathrm{SEQ}_{\mathrm{TAU}}$

- Sequents: $\Gamma = M_1, \cdots, M_s$ where $M_i \subset_{\mathrm{fin}} \mathbb{Z}_0 := \mathbb{Z} - \{0\}$ such that $(\forall a, b \in M_i)\,(a + b \neq 0)$
- *Axiom* $(\mathrm{A}_1): \ \{1\}, \{-1\}$
- *Weakening rules*

$$(\mathrm{W}_1): \ \frac{\Gamma}{M, \Gamma} \quad , \quad (\mathrm{W}_2): \ \frac{M \cup M', \Gamma}{M, \Gamma}$$

- *Substitution rule*

$$(\mathrm{S}): \ \frac{\Gamma}{\theta(\Gamma)}, \ \textit{where } \theta \in \mathit{Hom}(\mathrm{Seq} \to \mathrm{Seq})$$

- *Main rule*, where $\pm k \notin M_i, M_j', \Gamma$

$$(\mathrm{Q}): \ \frac{M_1, \cdots, M_r, \Gamma \qquad\qquad M_1', \cdots, M_{r'}', \Gamma}{\{k\} \cup M_1, \cdots, \{k\} \cup M_r, \{-k\} \cup M_1', \cdots, \{-k\} \cup M_{r'}', \Gamma}$$

- *Relation* $R := \{W_1, W_2, S\}^*$ (transitive closure)

$\mathrm{SEQ}_{\mathrm{TAU}}$: Some special cases

SEQ$_{\mathrm{TAU}}$: Some special cases

SEQ_{TAU}: Some special cases

- "Perfect" special case of weakening (W_0) :

$$\frac{\Gamma}{\{k\} \cup M_1, \cdots, \{k\} \cup M_r, \Gamma}$$

where $\pm k \notin M_i, \Gamma$

- "Perfect" special case of weakening (W_0) :

$$\frac{\Gamma}{\{k\} \cup M_1, \cdots, \{k\} \cup M_r, \Gamma}$$

where $\pm k \notin M_i, \Gamma$

- "Perfect" special case of Q whose side sequent (Γ) is empty, i.e. the following rule Q_0 :

$$\frac{M_1, \cdots, M_r \qquad\qquad M'_1, \cdots, M'_{r'}}{\{k\} \cup M_1, \cdots, \{k\} \cup M_r, \{-k\} \cup M'_1, \cdots, \{-k\} \cup M'_{r'}}$$

$$\textit{where } \left(\forall 1 \le i \le r, 1 \le j \le r'\right) \left(\pm k \notin M_i, M'_j\right)$$

$\mathrm{SEQ}_{\mathrm{TAU}}$: Examples

$\mathrm{SEQ}_{\mathrm{TAU}}$: Examples

Example

$$\dfrac{(S)\ \dfrac{\overset{(A_1)}{\{1\},\{-1\}} \quad \overset{(S)}{\rightarrowtail} \quad \{4\},\{-4\}}{\{2\},\{-2\} \qquad \{3\},\{-3,4\},\{-3,-4\}}\ (Q)}{\{1,2\},\{1,-2\},\{-1,3\},\{-1,-3,4\},\{-1,-3,-4\}}\ (Q)$$

SEQ$_{\mathrm{TAU}}$: Examples

Example

$$\dfrac{(S)\ \dfrac{\overset{(A_1)}{\{1\},\{-1\}} \qquad \overset{(S)}{\rightarrowtail} \qquad \{4\},\{-4\}}{\{2\},\{-2\} \qquad \{3\},\{-3,4\},\{-3,-4\}}\ (Q)}{\{1,2\},\{1,-2\},\{-1,3\},\{-1,-3,4\},\{-1,-3,-4\}}\ (Q)$$

Example

$$\dfrac{(Q_0)\ \dfrac{(S)\ \dfrac{\overset{(A_1)}{\{1\},\{-1\}}}{\{4\},\{-4\}}}{\{2\},\{-2,4\},\{-2,-4\}} \qquad \dfrac{\{2\},\{-2,4\},\{-2,-4\} \qquad \dfrac{\overset{(A_1)\overset{(S)}{\rightarrowtail}}{\{5\},\{-5\}}}{\{5\},\{-5\},\{-2,-4\}}\ (Ws)}{\{2,3\},\{-2,3,4\},\{-3,5\},\{-3,-5\},\{-2,-4\}}\ (Q)}{\{1,2\},\{1,-2,4\},\{-1,2,3\},\{-1,-2,3,4\},\{-1,-3,5\},\{-1,-3,-5\},\ \{-2,-4\}}\ (Q)$$

Semantics of $\mathrm{SEQ}_{\mathrm{TAU}}$

Definition

Semantics of $\mathrm{SEQ}_{\mathrm{TAU}}$

Definition

$$\Gamma = M_1, \cdots, M_s \hookrightarrow \varphi(\Gamma) := \bigvee_{i=1}^{s} \left(\bigwedge_{j \in M_i} \ell_j \right) \in DNF,$$

Semantics of $\mathrm{SEQ}_{\mathrm{TAU}}$

Definition

$$\Gamma = M_1, \cdots, M_s \hookrightarrow \varphi(\Gamma) := \bigvee_{i=1}^{s} \left(\bigwedge_{j \in M_i} \ell_j \right) \in DNF, \text{ where}$$

$$\ell_j := \begin{cases} x_j & \text{if } \; j > 0 \\ \neg x_{-j} & \text{if } \; j < 0 \end{cases} .$$

Semantics of $\mathrm{SEQ}_{\mathrm{TAU}}$

Definition

$$\Gamma = M_1, \cdots, M_s \hookrightarrow \varphi(\Gamma) := \bigvee_{i=1}^{s} \left(\bigwedge_{j \in M_i} \ell_j \right) \in DNF, \text{ where}$$

$$\ell_j := \begin{cases} x_j & \text{if } \; j > 0 \\ \neg x_{-j} & \text{if } \; j < 0 \end{cases}.$$

Denote by TAU the set of Γ such that $\varphi(\Gamma)$ is valid (as DNF).

Semantics of $\mathrm{SEQ}_{\mathrm{TAU}}$

Definition

$$\Gamma = M_1, \cdots, M_s \hookrightarrow \varphi(\Gamma) := \bigvee_{i=1}^{s} \left(\bigwedge_{j \in M_i} \ell_j \right) \in DNF, \text{ where}$$

$$\ell_j := \begin{cases} x_j & \text{if} \quad j > 0 \\ \neg x_{-j} & \text{if} \quad j < 0 \end{cases}.$$

Denote by TAU the set of Γ such that $\varphi(\Gamma)$ is valid (as DNF).

Theorem

Semantics of $\mathrm{SEQ}_{\mathrm{TAU}}$

Definition

$$\Gamma = M_1, \cdots, M_s \hookrightarrow \varphi(\Gamma) := \bigvee_{i=1}^{s} \left(\bigwedge_{j \in M_i} \ell_j \right) \in DNF, \text{ where}$$

$$\ell_j := \begin{cases} x_j & \text{if} \quad j > 0 \\ \neg x_{-j} & \text{if} \quad j < 0 \end{cases}.$$

Denote by TAU the set of Γ such that $\varphi(\Gamma)$ is valid (as DNF).

Theorem

1. *Γ is tree-like provable in SEQ_{TAU} iff Γ is dag-like provable in SEQ_{TAU}.*

Semantics of $\mathrm{SEQ}_{\mathrm{TAU}}$

Definition

$$\Gamma = M_1, \cdots, M_s \hookrightarrow \varphi(\Gamma) := \bigvee_{i=1}^{s} \left(\bigwedge_{j \in M_i} \ell_j \right) \in DNF, \text{ where}$$

$$\ell_j := \begin{cases} x_j & \text{if } \quad j > 0 \\ \neg x_{-j} & \text{if } \quad j < 0 \end{cases} .$$

Denote by TAU the set of Γ such that $\varphi(\Gamma)$ is valid (as DNF).

Theorem

1. *Γ is tree-like provable in SEQ_{TAU} iff Γ is dag-like provable in SEQ_{TAU}.*
2. *Γ is tree-like provable in SEQ_{TAU} iff $\Gamma \in TAU$.*

Semantics of $\mathrm{SEQ}_{\mathrm{TAU}}$

Definition

$$\Gamma = M_1, \cdots, M_s \hookrightarrow \varphi(\Gamma) := \bigvee_{i=1}^{s} \left(\bigwedge_{j \in M_i} \ell_j \right) \in DNF, \text{ where}$$

$$\ell_j := \begin{cases} x_j & \text{if} \quad j > 0 \\ \neg x_{-j} & \text{if} \quad j < 0 \end{cases} .$$

Denote by TAU the set of Γ such that $\varphi(\Gamma)$ is valid (as DNF).

Theorem

1. *Γ is tree-like provable in $\mathrm{SEQ}_{\mathrm{TAU}}$ iff Γ is dag-like provable in $\mathrm{SEQ}_{\mathrm{TAU}}$.*
2. *Γ is tree-like provable in $\mathrm{SEQ}_{\mathrm{TAU}}$ iff $\Gamma \in TAU$.*

Proof.

Easy. □

More on $\mathrm{SEQ}_{\mathrm{TAU}}$

More on $\mathrm{SEQ}_{\mathrm{TAU}}$

- Well-known "hard" tautologies have polynomial size dag-like proofs in $\mathrm{SEQ}_{\mathrm{TAU}}$ obtained by basic proof search (see below).

More on SEQ_{TAU}

- Well-known "hard" tautologies have polynomial size dag-like proofs in SEQ_{TAU} obtained by basic proof search (see below).

These examples include e.g.:

More on $\mathrm{SEQ}_{\mathrm{TAU}}$

- Well-known "hard" tautologies have polynomial size dag-like proofs in $\mathrm{SEQ}_{\mathrm{TAU}}$ obtained by basic proof search (see below).

These examples include e.g.:

1. Doubling names tautologies by Takeuti and Statman.

More on $\mathrm{SEQ}_{\mathrm{TAU}}$

- Well-known "hard" tautologies have polynomial size dag-like proofs in $\mathrm{SEQ}_{\mathrm{TAU}}$ obtained by basic proof search (see below).

These examples include e.g.:

1. Doubling names tautologies by Takeuti and Statman.
2. Fibonacci-style tautology by Haeusler and Pereira.

More on $\mathrm{SEQ}_{\mathrm{TAU}}$

- Well-known "hard" tautologies have polynomial size dag-like proofs in $\mathrm{SEQ}_{\mathrm{TAU}}$ obtained by basic proof search (see below).

These examples include e.g.:

1. Doubling names tautologies by Takeuti and Statman.
2. Fibonacci-style tautology by Haeusler and Pereira.
3. Pigeonhole principle.

More on $\mathrm{SEQ}_{\mathrm{TAU}}$

- Well-known "hard" tautologies have polynomial size dag-like proofs in $\mathrm{SEQ}_{\mathrm{TAU}}$ obtained by basic proof search (see below).

These examples include e.g.:

1. Doubling names tautologies by Takeuti and Statman.
2. Fibonacci-style tautology by Haeusler and Pereira.
3. Pigeonhole principle.
4. Clique coloring principle (k-clique tautology).

More on $\mathrm{SEQ}_{\mathrm{TAU}}$

- Well-known "hard" tautologies have polynomial size dag-like proofs in $\mathrm{SEQ}_{\mathrm{TAU}}$ obtained by basic proof search (see below).

These examples include e.g.:

1. Doubling names tautologies by Takeuti and Statman.
2. Fibonacci-style tautology by Haeusler and Pereira.
3. Pigeonhole principle.
4. Clique coloring principle (k-clique tautology).

Hence neither resolution nor cutting planes p-simulate $\mathrm{SEQ}_{\mathrm{TAU}}$.

- Well-known "hard" tautologies have polynomial size dag-like proofs in $\mathrm{SEQ}_{\mathrm{TAU}}$ obtained by basic proof search (see below).

These examples include e.g.:

1. Doubling names tautologies by Takeuti and Statman.
2. Fibonacci-style tautology by Haeusler and Pereira.
3. Pigeonhole principle.
4. Clique coloring principle (k-clique tautology).

Hence neither resolution nor cutting planes p-simulate $\mathrm{SEQ}_{\mathrm{TAU}}$.

Theorem

More on $\mathrm{SEQ}_{\mathrm{TAU}}$

- Well-known "hard" tautologies have polynomial size dag-like proofs in $\mathrm{SEQ}_{\mathrm{TAU}}$ obtained by basic proof search (see below).

These examples include e.g.:

1. Doubling names tautologies by Takeuti and Statman.
2. Fibonacci-style tautology by Haeusler and Pereira.
3. Pigeonhole principle.
4. Clique coloring principle (k-clique tautology).

Hence neither resolution nor cutting planes p-simulate $\mathrm{SEQ}_{\mathrm{TAU}}$.

Theorem

There are $\Gamma \in$ TAU such that for all tree-like deductions T of Γ, $\#T$ is exponential in $\#\Gamma$, whereas $\delta(\Gamma)$ is polynomial in $\#\Gamma$.

Reminder: Clique coloring principle

Reminder: Clique coloring principle

Theorem

Reminder: Clique coloring principle

Theorem

Clique coloring principle: *No n-element graph G, $|G| = n$, has a $(k-1)$-colored k-element clique $K \subseteq G$ such that $2 \leq k = |K| \leq n$ and there is no edge (in G) between any pair of vertices (in K) having the same color.*

Basic dag-like proof search in SEQ_{TAU}

Basic dag-like proof search in $\mathrm{SEQ}_{\mathrm{TAU}}$

Consider any given sequent Γ_0. Starting with Γ_0 reduce sequents by inverting the rules (W_0) and (Q) repeatedly, while simultaneously analyzing pairs of new sequents Γ_i, Γ_j thus obtained which are not axioms and occur in different branches:

Basic dag-like proof search in $\mathrm{SEQ}_{\mathrm{TAU}}$

Consider any given sequent Γ_0. Starting with Γ_0 reduce sequents by inverting the rules (W_0) and (Q) repeatedly, while simultaneously analyzing pairs of new sequents Γ_i, Γ_j thus obtained which are not axioms and occur in different branches:

1. If $\{1\}, \{-1\} R \Gamma_i$ (resp. $\{1\}, \{-1\} R \Gamma_j$), then add arrow $(\mathrm{A}_1) \to \Gamma_i$ (resp. $(\mathrm{A}_1) \to \Gamma_j$) and close the corresponding branch.

Basic dag-like proof search in $\mathrm{SEQ}_{\mathrm{TAU}}$

Consider any given sequent Γ_0. Starting with Γ_0 reduce sequents by inverting the rules (W_0) and (Q) repeatedly, while simultaneously analyzing pairs of new sequents Γ_i, Γ_j thus obtained which are not axioms and occur in different branches:

1. If $\{1\}, \{-1\} R\Gamma_i$ (resp. $\{1\}, \{-1\} R\Gamma_j$), then add arrow $(\mathrm{A}_1) \to \Gamma_i$ (resp. $(\mathrm{A}_1) \to \Gamma_j$) and close the corresponding branch.
2. If $\Gamma_i R \Gamma_j$ (resp. $\Gamma_j R \Gamma_i$), then add arrow $\Gamma_i \to \Gamma_j$ (resp. $\Gamma_j \to \Gamma_i$) and don't reduce Γ_j (resp. Γ_i) anymore.

Basic dag-like proof search in $\mathrm{SEQ}_{\mathrm{TAU}}$

Consider any given sequent Γ_0. Starting with Γ_0 reduce sequents by inverting the rules (W_0) and (Q) repeatedly, while simultaneously analyzing pairs of new sequents Γ_i, Γ_j thus obtained which are not axioms and occur in different branches:

1. If $\{1\}, \{-1\} R \Gamma_i$ (resp. $\{1\}, \{-1\} R \Gamma_j$), then add arrow $(A_1) \rightarrow \Gamma_i$ (resp. $(A_1) \rightarrow \Gamma_j$) and close the corresponding branch.
2. If $\Gamma_i R \Gamma_j$ (resp. $\Gamma_j R \Gamma_i$), then add arrow $\Gamma_i \rightarrow \Gamma_j$ (resp. $\Gamma_j \rightarrow \Gamma_i$) and don't reduce Γ_j (resp. Γ_i) anymore.

This reduction procedure terminates. Consider the resulting sequent dag D and let $D \rhd_0 D'$.
If all leaves of D are axioms, then D' is a desired dag-like deduction of Γ. Otherwise Γ is invalid.

Dag-compression vs CUT

Dag-compression vs CUT

- It is well-known that adding *cut rule* to cutfree proof systems can exponentially accelerate propositional provability (re: propositional *speed-up*). However proof systems with (CUT) or *modus ponens* or similar non-analytic inferences, known as general *Frege systems*, don't admit reasonable poor search.

Dag-compression vs CUT

- It is well-known that adding *cut rule* to cutfree proof systems can exponentially accelerate propositional provability (re: propositional *speed-up*). However proof systems with (CUT) or *modus ponens* or similar non-analytic inferences, known as general *Frege systems*, don't admit reasonable poor search.
- Dag-like cutfree calculus $\mathrm{SEQ}_{\mathrm{TAU}}$ shows that adding dag-like *substitution rules* provides analogous acceleration of provability (either by dag-compression or direct proof search) – in all most familiar cases of cut-like speed-up.
  But $\mathrm{SEQ}_{\mathrm{TAU}}$ preserves good proof search options.

Dag-compression vs CUT

- It is well-known that adding *cut rule* to cutfree proof systems can exponentially accelerate propositional provability (re: propositional *speed-up*). However proof systems with (CUT) or *modus ponens* or similar non-analytic inferences, known as general *Frege systems*, don't admit reasonable poor search.
- Dag-like cutfree calculus $\mathrm{SEQ}_{\mathrm{TAU}}$ shows that adding dag-like *substitution rules* provides analogous acceleration of provability (either by dag-compression or direct proof search) – in all most familiar cases of cut-like speed-up. But $\mathrm{SEQ}_{\mathrm{TAU}}$ preserves good proof search options.
- By familiar cut-elimination arguments, any Frege system is reducible to tree-like, and hence also dag-like version of $\mathrm{SEQ}_{\mathrm{TAU}}$ without substitution.

Dag-compression vs CUT

- It is well-known that adding *cut rule* to cutfree proof systems can exponentially accelerate propositional provability (re: propositional *speed-up*). However proof systems with (CUT) or *modus ponens* or similar non-analytic inferences, known as general *Frege systems*, don't admit reasonable poor search.
- Dag-like cutfree calculus $\mathrm{SEQ}_{\mathrm{TAU}}$ shows that adding dag-like *substitution rules* provides analogous acceleration of provability (either by dag-compression or direct proof search) – in all most familiar cases of cut-like speed-up.
  But $\mathrm{SEQ}_{\mathrm{TAU}}$ preserves good proof search options.
- By familiar cut-elimination arguments, any Frege system is reducible to tree-like, and hence also dag-like version of $\mathrm{SEQ}_{\mathrm{TAU}}$ without substitution. Can analogous cut elimination with substitution be done with sub-exponential growth of the resulting dag-like deductions in $\mathrm{SEQ}_{\mathrm{TAU}}$?

"Academic" Conjectures C1, C2

“Academic” Conjectures C1, C2

Definition

“Academic” Conjectures C1, C2

Definition

1. “Academic” Conjecture C1: For every $\Gamma \in$ TAU, $\delta(\Gamma)$ is polynomial in $\#\Gamma$.

“Academic” Conjectures C1, C2

Definition

1. “Academic” Conjecture C1: For every $\Gamma \in$ TAU, $\delta(\Gamma)$ is polynomial in $\#\Gamma$.
2. “Academic” Conjecture C2: Every $\Gamma \in$ TAU has dag-like $\mathrm{SEQ}_{\mathrm{TAU}}$-deduction D such that $\#D$ is polynomial in $\#\Gamma$.

Definition

1. "Academic" Conjecture C1: For every $\Gamma \in$ TAU, $\delta(\Gamma)$ is polynomial in $\#\Gamma$.
2. "Academic" Conjecture C2: Every $\Gamma \in$ TAU has dag-like SEQ_{TAU}-deduction D such that $\#D$ is polynomial in $\#\Gamma$.

Theorem

“Academic” Conjectures C1, C2

Definition

1. “Academic” Conjecture C1: For every $\Gamma \in$ TAU, $\delta(\Gamma)$ is polynomial in $\#\Gamma$.
2. “Academic” Conjecture C2: Every $\Gamma \in$ TAU has dag-like SEQ$_{\mathrm{TAU}}$-deduction D such that $\#D$ is polynomial in $\#\Gamma$.

Theorem

1. *C1 implies C2.*

“Academic” Conjectures C1, C2

Definition

1. “Academic” Conjecture C1: For every $\Gamma \in \mathrm{TAU}$, $\delta(\Gamma)$ is polynomial in $\#\Gamma$.
2. “Academic” Conjecture C2: Every $\Gamma \in \mathrm{TAU}$ has dag-like $\mathrm{SEQ}_{\mathrm{TAU}}$-deduction D such that $\#D$ is polynomial in $\#\Gamma$.

Theorem

1. *C1 implies C2.*
2. *C2 implies* $\mathrm{NP} = \mathrm{coNP}$.

"Academic" Conjectures C1, C2

Definition

1. "Academic" Conjecture C1: For every $\Gamma \in \mathrm{TAU}$, $\delta(\Gamma)$ is polynomial in $\#\Gamma$.
2. "Academic" Conjecture C2: Every $\Gamma \in \mathrm{TAU}$ has dag-like $\mathrm{SEQ}_{\mathrm{TAU}}$-deduction D such that $\#D$ is polynomial in $\#\Gamma$.

Theorem

1. *C1 implies C2.*
2. *C2 implies* $\mathrm{NP} = \mathrm{coNP}$.

Proof.

Clear. □

Appendix: P-NP connections

Appendix: P-NP connections

Definition

Let $\mathrm{SEQ}^{0}_{\mathrm{TAU}}$ be subsystem of $\mathrm{SEQ}_{\mathrm{TAU}}$ that includes only special case (Q_0) of the main rule in which side sequent $\Gamma = \emptyset$.

Definition

Let $\mathrm{SEQ}^{0}_{\mathrm{TAU}}$ be subsystem of $\mathrm{SEQ}_{\mathrm{TAU}}$ that includes only special case (Q_0) of the main rule in which side sequent $\Gamma = \emptyset$.
Let $\mathrm{TAU}^{(n)}_{0}$ be the set of sequents with at most $n+1$ clauses with at most n literals in each clause, which are derivable in $\mathrm{SEQ}^{0}_{\mathrm{TAU}}$.

Appendix: P-NP connections

Definition

Let $\mathrm{SEQ}^0_{\mathrm{TAU}}$ be subsystem of $\mathrm{SEQ}_{\mathrm{TAU}}$ that includes only special case (Q_0) of the main rule in which side sequent $\Gamma = \emptyset$.
Let $\mathrm{TAU}^{(n)}_0$ be the set of sequents with at most $n+1$ clauses with at most n literals in each clause, which are derivable in $\mathrm{SEQ}^0_{\mathrm{TAU}}$.

Lemma

$TAU^{(n)}_0 \in$ NP.

Definition

Appendix: P-NP connections

Definition

Let $\mathrm{SEQ}^0_{\mathrm{TAU}}$ be subsystem of $\mathrm{SEQ}_{\mathrm{TAU}}$ that includes only special case (Q_0) of the main rule in which side sequent $\Gamma = \emptyset$.
Let $\mathrm{TAU}^{(n)}_0$ be the set of sequents with at most $n+1$ clauses with at most n literals in each clause, which are derivable in $\mathrm{SEQ}^0_{\mathrm{TAU}}$.

Lemma

$TAU^{(n)}_0 \in \mathrm{NP}$.

Definition

Plausible Conjecture C3:
$\mathrm{TAU}^{(n)}_0$ is not representable in a certain concrete (simple) algebra $\mathfrak{A}_n$ by a term whose length is polynomial in n.

Appendix: P-NP connections

Definition

Let $\mathrm{SEQ}^0_{\mathrm{TAU}}$ be subsystem of $\mathrm{SEQ}_{\mathrm{TAU}}$ that includes only special case (Q_0) of the main rule in which side sequent $\Gamma = \emptyset$.
Let $\mathrm{TAU}^{(n)}_0$ be the set of sequents with at most $n+1$ clauses with at most n literals in each clause, which are derivable in $\mathrm{SEQ}^0_{\mathrm{TAU}}$.

Lemma

$TAU^{(n)}_0 \in \mathrm{NP}$.

Definition

Plausible Conjecture C3:
$\mathrm{TAU}^{(n)}_0$ is not representable in a certain concrete (simple) algebra $\mathfrak{A}_n$ by a term whose length is polynomial in n.

Theorem

C3 implies $\mathrm{P} < \mathrm{NP}$.