ABSTRACT

A description is an entity that can be interpreted as true or false of an object, and using feature structures as descriptions accrues several computational benefits. In this paper, I create an explicit interpretation of a typed feature structure used as a description, define the notion of a satisfiable feature structure, and create a simple and effective algorithm to decide if a feature structure is satisfiable.

1. INTRODUCTION

Describing objects is one of several purposes for which linguists use feature structures. A description is an entity that can be interpreted as true or false of an object. For example, the conventional interpretation of the description ‘it is black’ is true of a soot particle, but false of a snowflake. Therefore, any use of a feature structure to describe an object demands that the feature structure can be interpreted as true or false of the object. In this paper, I tailor the semantics of [King 1989] to suit the typed feature structures of [Carpenter 1992], and create an explicit interpretation of a typed feature structure used as a description. I then use this interpretation to define the notion of a satisfiable feature structure.

Though no feature structure algebra provides descriptions as expressive as those provided by a feature logic, using feature structures to describe objects profits from a large stock of available computational techniques to represent, test and process feature structures. In this paper, I demonstrate the computational benefits of marrying a tractable syntax and an explicit semantics by creating a simple and effective algorithm to decide the satisfiability of a feature structure. Gerdemann and Götz’s Troll type resolution system implements both the semantics and an efficient refinement of the satisfiability algorithm I present here (see [Götz 1993], [Gerdemann and King 1994] and [Gerdemann (fc)]).

2. A FEATURE STRUCTURE SEMANTICS

A signature provides the symbols from which to construct typed feature structures, and an interpretation gives those symbols meaning.

Definition 1. Σ is a signature iff

\[ \Sigma = \langle \langle \mathcal{Q}, \mathcal{T}, \leq, \mathcal{S}, \mathfrak{A}, \mathfrak{F} \rangle, Q \text{ is a set,} \]

\[ \langle \mathcal{T}, \leq \rangle \text{ is a partial order,} \]

\[ \mathcal{S} = \{ \sigma \in \mathcal{T} \mid \text{for each } \tau \in \mathcal{T}, \]

\[ \text{if } \sigma \preceq \tau \text{ then } \sigma = \tau \} \}

\[ \mathfrak{A} \text{ is a set,} \]

\[ \mathfrak{F} \text{ is a partial function from the Cartesian product of } \mathcal{T} \text{ and } \mathfrak{A} \text{ to } \mathcal{T}, \]

\[ \text{and for each } \tau \in \mathcal{T}, \text{ each } \tau' \in \mathcal{T} \text{ and each } \alpha \in \mathfrak{A}, \]

\[ \text{if } \mathfrak{F}(\tau, \alpha) \text{ is defined and } \tau \preceq \tau' \]

\[ \text{then } \mathfrak{F}(\tau', \alpha) \text{ is defined, and} \]

\[ \mathfrak{F}(\tau, \alpha) \preceq \mathfrak{F}(\tau', \alpha). \]

Henceforth, I tacitly work with a signature \( \langle \mathcal{Q}, \mathcal{T}, \leq, \mathcal{S}, \mathfrak{A}, \mathfrak{F} \rangle \). I call members of \( \mathcal{Q} \) states, members of \( \mathcal{T} \) types, \( \preceq \) subsumption, members of \( \mathcal{S} \) species, members of \( \mathfrak{A} \) attributes, and \( \mathfrak{F} \) appropriateness.

Definition 2. \( I \) is an interpretation iff

\[ I \text{ is a triple } \langle U, S, A \rangle, \]

\[ U \text{ is a set,} \]

\[ S \text{ is a total function from } U \text{ to } \mathcal{S} \]

\[ A \text{ is a total function from } \mathfrak{A} \text{ to the set of partial functions from } U \text{ to } U, \]

\[ \text{for each } \alpha \in \mathfrak{A} \text{ and each } u \in U, \]

\[ \text{if } A(\alpha)(u) \text{ is defined} \]

\[ \text{then } \mathfrak{F}(S(u), \alpha) \text{ is defined, and} \]

\[ \mathfrak{F}(S(u), \alpha) \preceq S(A(\alpha)(u)), \]

\[ \text{and for each } \alpha \in \mathfrak{A} \text{ and each } u \in U, \]

\[ \text{if } \mathfrak{F}(S(u), \alpha) \text{ is defined} \]

\[ \text{then } A(\alpha)(u) \text{ is defined.} \]

Suppose that \( I \) is an interpretation \( \langle U, S, A \rangle \).

I call each member of \( U \) an object in \( I \).

Each type denotes a set of objects in \( I \). The
denotations of the species partition \(U\), and \(S\) assigns each object in \(I\) the unique species whose denotation contains the object: object \(u\) is in the denotation of species \(\sigma\) iff \(\sigma = S(u)\). Subsumption encodes a relationship between the denotations of species and types: object \(u\) is in the denotation of type \(\tau\) iff \(\tau \subseteq S(u)\). So, if \(\tau_1 \subseteq \tau_2\) then the denotation of type \(\tau_1\) contains the denotation of type \(\tau_2\).

Each attribute denotes a partial function from the objects in \(I\) to the objects in \(J\), and \(A\) assigns each attribute the partial function it denotes. Appropriateness encodes a relationship between the denotations of species and attributes: if \(\mathcal{G}(\sigma, \alpha)\) is defined then the denotation of attribute \(\alpha\) acts upon each object in the denotation of species \(\sigma\) to yield an object in the denotation of type \(\tau\), but if \(\mathcal{G}(\sigma, \alpha)\) is undefined then the denotation of attribute \(\alpha\) acts upon no object in the denotation of species \(\sigma\). So, if \(\mathcal{G}(\tau, \alpha)\) is defined then the denotation of attribute \(\alpha\) acts upon each object in the denotation of type \(\tau\) to yield an object in the denotation of type \(\mathcal{G}(\tau, \alpha)\).

I call a finite sequence of attributes a path, and write \(\mathfrak{P}\) for the set of paths.

**Definition 3.** \(P\) is the path interpretation function under \(I\) iff

- \(I\) is an interpretation \(\langle U, S, A \rangle\),
- \(P\) is a total function from \(\mathfrak{P}\) to the set of partial functions from \(U \to U\), and for each \(\langle \alpha_1, \ldots, \alpha_n \rangle \in \mathfrak{P}\),
- \(P(\alpha_1, \ldots, \alpha_n)\) is the functional composition of \(A(\alpha_1), \ldots, A(\alpha_n)\).

I write \(P_I\) for the path interpretation function under \(I\).

**Definition 4.** \(F\) is a feature structure iff

- \(F\) is a quadruple \((Q, q, \delta, \theta)\),
- \(Q\) is a finite subset of \(\mathcal{G}\),
- \(q \in Q\),
- \(\delta\) is a finite partial function from the Cartesian product of \(Q\) and \(\mathfrak{A}\) to \(Q\),
- \(\theta\) is a total function from \(Q\) to \(\mathcal{T}\), and for each \(q' \in Q\),
  - for some \(\pi \in \mathfrak{P}\), \(\pi\) runs to \(q'\) in \(F\) iff \(\langle \alpha_1, \ldots, \alpha_n \rangle \in \mathfrak{P}\), \(q' \in Q\), and for some \(\{q_0, \ldots, q_n\} \subseteq Q\),
    - \(q = q_0\),
    - for each \(i < n\), \(\delta(q_i, \alpha_{i+1})\) is defined, and \(\delta(q_i, \alpha_{i+1}) = q_{i+1}\), and \(q_n = q'\).

Each feature structure is a connected Moore machine (see [Moore 1956]) with finitely many states, input alphabet \(\mathfrak{A}\), and output alphabet \(\mathcal{T}\).

**Definition 5.** \(F\) is true of \(u\) under \(I\) iff

- \(F\) is a feature structure \((Q, q, \delta, \theta)\),
- \(I\) is an interpretation \(\langle U, S, A \rangle\),
- \(u\) is an object in \(I\), and for each \(\pi_1 \in \mathfrak{P}\), each \(\pi_2 \in \mathfrak{P}\) and each \(q' \in Q\),
  - if \(\pi_1\) runs to \(q'\) in \(F\), and \(\pi_2\) runs to \(q'\) in \(F\),
  - then \(P_I(\pi_1)(u)\) is defined, \(P_I(\pi_2)(u)\) is defined, \(P_I(\pi_1)(u) = P_I(\pi_2)(u)\), and \(\theta(q') \subseteq S(P_I(\pi_1)(u))\).

**Definition 6.** \(F\) is a satisfiable feature structure iff

- \(F\) is a feature structure, and
- for some interpretation \(I\) and some object \(u\) in \(I\), \(F\) is true of \(u\) under \(I\).

### 3. MORPHS

The abundance of interpretations seems to preclude an effective algorithm to decide if a feature structure is satisfiable. However, I insert morphs between feature structures and objects to yield an interpretation free characterisation of a satisfiable feature structure.

**Definition 7.** \(M\) is a semi-morph iff

- \(M\) is a triple \(\langle \Delta, \Gamma, \Lambda \rangle\),
- \(\Delta\) is a nonempty subset of \(\mathfrak{P}\),
- \(\Gamma\) is an equivalence relation over \(\Delta\),
- \(\Lambda\) is a total function from \(\Delta\) to \(\mathcal{G}\),
- for each \(\alpha \in \mathfrak{A}\), each \(\pi_1 \in \mathfrak{P}\) and each \(\pi_2 \in \mathfrak{P}\),
  - if \(\pi_1 \alpha \in \Delta\) and \((\pi_1, \pi_2) \in \Gamma\), then \(\langle \pi_1 \alpha, \pi_2 \alpha \rangle \in \Gamma\),

- \(\Lambda\) is a total function from \(\Delta\) to \(\mathcal{G}\),
- \(\Lambda\) is a total function from \(\Delta\) to \(\mathcal{G}\),
- for each \(\pi_1 \in \mathfrak{P}\) and each \(\pi_2 \in \mathfrak{P}\),
- if \(\pi_1 \alpha \in \Delta\) and \(\pi_2 \beta \in \Delta\), then \(\Lambda(\pi_1) = \Lambda(\pi_2)\), and for each \(\alpha \in \mathfrak{A}\) and each \(\pi \in \mathfrak{P}\),
  - if \(\pi \alpha \in \Delta\), then \(\pi \in \Delta\), \(\mathcal{G}(\Lambda(\pi), \alpha)\) is defined, and \(\mathcal{G}(\Lambda(\pi), \alpha) \subseteq \Lambda(\pi \alpha)\).

**Definition 8.** \(M\) is a morph iff

- \(M\) is a semi-morph \(\langle \Delta, \Gamma, \Lambda \rangle\), and
- for each \(\alpha \in \mathfrak{A}\) and each \(\pi \in \mathfrak{P}\),
  - if \(\pi \in \Delta\) and \(\mathcal{G}(\Lambda(\pi), \alpha)\) is defined
    - then \(\pi \alpha \in \Delta\).

Each morph is the Mosher abstraction (see [Mosher 1988]) of a connected and totally well-typed (see [Carpenter 1992]) Moore machine with possibly infinitely many states, input alphabet \(\mathfrak{A}\), and output alphabet \(\mathcal{G}\).
Definition 9. $M$ abstracts $u$ under $I$ iff

$M$ is a morph $(\Delta, \Gamma, \Lambda)$, $I$ is an interpretation $(U, S, A)$, $u$ is an object in $I$, for each $\pi_1 \in \mathfrak{P}$ and each $\pi_2 \in \mathfrak{P}$, $(\pi_1, \pi_2) \in \Gamma$ iff $P_I(\pi_1)(u)$ is defined, $P_I(\pi_2)(u)$ is defined, and $P_I(\pi_1)(u) = P_I(\pi_2)(u)$, and for each $\sigma \in S$ and each $\pi \in \mathfrak{P}$, $(\pi, \sigma) \in \Lambda$ iff $P_I(\pi)(u)$ is defined, and $\sigma = S(P_I(\pi)(u))$.

Proposition 10. For each interpretation $I$ and each object $u$ in $I$,

some unique morph abstracts $u$ under $I$. I thus write of the abstraction of $u$ under $I$.

Definition 11. $u$ is a standard object iff

$u$ is a quadruple $(\Delta, \Gamma, \Lambda, E)$, $(\Delta, \Gamma, \Lambda)$ is a morph, and $E$ is an equivalence class under $\Gamma$.

I write $\bar{U}$ for the set of standard objects, write $\bar{S}$ for the total function from $\bar{U}$ to $S$, where for each $\sigma \in S$ and each $(\Delta, \Gamma, \Lambda, E) \in \bar{U}$, 

$\bar{S}(\Delta, \Gamma, \Lambda, E) = \sigma$

iff for some $\pi \in E$, $\Lambda(\pi) = \sigma$,

and write $\bar{A}$ for the total function from $\mathfrak{A}$ to the set of partial functions from $\bar{U}$ to $\bar{U}$, where for each $\alpha \in \mathfrak{A}$, each $(\Delta, \Gamma, \Lambda, E) \in \bar{U}$ and each $(\Delta', \Gamma', \Lambda', E') \in \bar{U}$,

$\bar{A}(\alpha)(\Delta, \Gamma, \Lambda, E) \text{ is defined, and }$

$\bar{A}(\alpha)(\Delta, \Gamma, \Lambda, E) = (\Delta', \Gamma', \Lambda', E')$

iff $(\Delta, \Gamma, \Lambda) = (\Delta', \Gamma', \Lambda')$, and for some $\pi \in E$, $\pi\alpha \in E'$.

Lemma 12. $(\bar{U}, \bar{S}, \bar{A})$ is an interpretation.

I write $\bar{I}$ for $(\bar{U}, \bar{S}, \bar{A})$.

Lemma 13. For each $(\Delta, \Gamma, \Lambda, E) \in \bar{U}$, each $(\Delta', \Gamma', \Lambda', E') \in \bar{U}$ and each $\pi \in \mathfrak{P}$,

$P_{\bar{I}}(\pi)(\Delta, \Gamma, \Lambda, E) \text{ is defined, and }$

$P_{\bar{I}}(\pi)(\Delta, \Gamma, \Lambda, E) = (\Delta', \Gamma', \Lambda', E')$

iff $(\Delta, \Gamma, \Lambda) = (\Delta', \Gamma', \Lambda')$, and for some $\pi' \in E$, $\pi'\pi \in E'$.

Proof. By induction on the length of $\pi$. ■

Lemma 14. For each $(\Delta, \Gamma, \Lambda, E) \in \bar{U}$, if $E$ is the equivalence class of the empty path under $\Gamma$ then the abstraction of $(\Delta, \Gamma, \Lambda, E)$ under $\bar{I}$ is $(\Delta, \Gamma, \Lambda)$.

Proposition 15. For each morph $M$, for some interpretation $I$ and some object $u$ in $I$,

$M$ is the abstraction of $u$ under $I$. 
Definition 16. $F$ approximates $M$ iff $F$ is a feature structure $(Q, q, \delta, \theta)$, $M$ is a morph $(\Delta, \Gamma, \Lambda)$, and for each $\pi_1 \in \Psi$, each $\pi_2 \in \Psi$ and each $q' \in Q$, if $\pi_1$ runs to $q'$ in $F$, and $\pi_2$ runs to $q'$ in $F$ then $(\pi_1, \pi_2) \in \Gamma$, and
$\theta(q') \leq \Lambda(\pi_1)$.

A feature structure approximates a morph iff the Mosher abstraction of the feature structure abstractly subsumes (see [CARPENTER 1992]) the morph.

Proposition 17. For each interpretation $I$, each object $u$ in $I$ and each feature structure $F$,

$F$ is true of $u$ under $I$ iff $F$ approximates the abstraction of $u$ under $I$.

Theorem 18. For each feature structure $F$, $F$ is satisfiable iff $F$ approximates some morph.

Proof. From propositions 15 and 17.  

4. RESOLVED FEATURE STRUCTURES

Though theorem 15 gives an interpretation free characterisation of a satisfiable feature structure, the characterisation still seems to admit of no effective algorithm to decide if a feature structure is satisfiable. However, I use theorem 18 and resolved feature structures to yield a less general interpretation free characterisation of a satisfiable feature structure that admits of such an algorithm.

Definition 19. $R$ is a resolved feature structure iff

$R$ is a feature structure $(Q, q, \delta, \rho)$, $\rho$ is a total function from $Q$ to $\mathcal{S}$, and for each $\alpha \in \mathfrak{A}$ and each $q' \in Q$, if $\delta(q', \alpha)$ is defined

then $\mathfrak{s}(\rho(q'), \alpha)$ is defined, and $\mathfrak{s}(\rho(q'), \alpha) \leq \rho(\delta(q', \alpha))$.

Each resolved feature structure is a well-typed (see [CARPENTER 1992]) feature structure with output alphabet $\mathcal{S}$.

Definition 20. $R$ is a resolvent of $F$ iff

$R$ is a resolved feature structure $(Q, q, \delta, \rho)$, $F$ is a feature structure $(Q, q, \delta, \theta)$, and for each $q' \in Q$, $\theta(q') \leq \rho(q')$.

Proposition 21. For each interpretation $I$, each object $u$ in $I$ and each feature structure $F$,

$F$ is true of $u$ under $I$ iff some resolvent of $F$ is true of $u$ under $I$.

Definition 22. $(\Omega, \Xi, \preceq, \mathcal{S}, \mathfrak{A}, \mathfrak{s})$ is rational iff for each $\sigma \in \mathcal{S}$ and each $\alpha \in \mathfrak{A}$, if $\mathfrak{s}(\sigma, \alpha)$ is defined then for some $\sigma' \in \mathcal{S}$, $\mathfrak{s}(\sigma, \alpha) \preceq \sigma'$.

Proposition 23. If $(\Omega, \Xi, \preceq, \mathcal{S}, \mathfrak{A}, \mathfrak{s})$ is rational then for each resolved feature structure $R$, $R$ is satisfiable.

Proof. Suppose that $R = (Q, q, \delta, \rho)$ and $\beta$ is a bijection from ordinal $\zeta$ to $\mathcal{S}$. Let

$\Delta_0 = \{ \pi \mid \pi \text{ runs to } q' \text{ in } R \}$, $\Gamma_0 = \{ (\pi_1, \pi_2) \mid \pi_1 \text{ runs to } q' \text{ in } R, \pi_2 \text{ runs to } q' \text{ in } R \}$, $\Lambda_0 = \{ (\pi, \sigma) \mid \pi \text{ runs to } q' \text{ in } R \}$.

For each $n \in \mathbb{N}$, let

$\Delta_{n+1} = \Delta_n \cup \{ \alpha \in \mathfrak{A}, \pi \in \Delta_n, \text{ and } \mathfrak{s}(\Lambda_n(\pi), \alpha) \text{ is defined} \}$,

$\Gamma_{n+1} = \Gamma_n \cup \{ \pi \in \Delta_n, \pi_1 \alpha \in \Delta_{n+1}, \pi_2 \alpha \in \Delta_{n+1}, \text{ and } (\pi_1, \pi_2) \in \Gamma_n \}$,

$\Lambda_{n+1} = \{ \alpha \in \mathfrak{A}, \pi \in \Delta_n, \pi \alpha \in \mathcal{S} \}$,

$\Lambda_n \cup \{ (\pi \alpha, \beta(\xi)) \mid \pi \alpha \in \Delta_{n+1}, \text{ and } \xi \text{ is the least ordinal in } \zeta \text{ such that } \mathfrak{s}(\Lambda_n(\pi), \alpha) \leq \beta(\xi) \}$.

For each $n \in \mathbb{N}$, $(\Delta_n, \Gamma_n, \Lambda_n)$ is a semi-morph. Let

$\Delta = \bigcup \{ \Delta_n \mid n \in \mathbb{N} \}$, $\Gamma = \bigcup \{ \Gamma_n \mid n \in \mathbb{N} \}$, and $\Lambda = \bigcup \{ \Lambda_n \mid n \in \mathbb{N} \}$.

$(\Delta, \Gamma, \Lambda)$ is a morph that $R$ approximates. By theorem 18, $R$ is satisfiable.  

Theorem 24. If $(\Omega, \Xi, \preceq, \mathcal{S}, \mathfrak{A}, \mathfrak{s})$ is rational then for each feature structure $F$,

$F$ is satisfiable iff $F$ has a resolvent.

Proof. From propositions 21 and 23.  

5. A SATISFIABILITY ALGORITHM

In this section, I use theorem 24 to show how – given a rational signature that meets reasonable computational conditions – to construct an effective algorithm to decide if a feature structure is satisfiable.
Definition 25. \( (\Omega, \mathcal{I}, \preceq, \mathcal{S}, \mathcal{A}, \mathcal{F}) \) is computable iff
\( \Omega, \mathcal{I} \) and \( \mathcal{A} \) are countable,
\( \mathcal{S} \) is finite,
for some effective function \( \text{SUB} \),
for each \( \tau_1 \in \mathcal{I} \) and each \( \tau_2 \in \mathcal{I} \),
if \( \tau_1 \preceq \tau_2 \)
then \( \text{SUB}(\tau_1, \tau_2) = \text{‘true’} \)
otherwise \( \text{SUB}(\tau_1, \tau_2) = \text{‘false’} \), and
for some effective function \( \text{APP} \),
for each \( \tau \in \mathcal{I} \) and each \( \alpha \in \mathcal{A} \),
if \( \mathcal{F}(\tau, \alpha) \) is defined
then \( \text{APP}(\tau, \alpha) = \mathcal{F}(\tau, \alpha) \)
otherwise \( \text{APP}(\tau, \alpha) = \text{‘undefined’} \).

Proposition 26. If \( (\Omega, \mathcal{I}, \preceq, \mathcal{S}, \mathcal{A}, \mathcal{F}) \) is computable then for some effective function \( \text{RES} \),
for each feature structure \( F \),
\( \text{RES}(F) = \{ \text{the resolvants of } F \} \).

Proof. Since \( (\Omega, \mathcal{I}, \preceq, \mathcal{S}, \mathcal{A}, \mathcal{F}) \) is computable,
for some effective function \( \text{GEN} \),
for each finite \( Q \subseteq \Omega \),
\( \text{GEN}(Q) = \{ \text{the total functions from } Q \text{ to } S \} \),
for some effective function \( \text{TEST}_1 \),
for each finite set \( Q \), each finite partial function \( \delta \) from the Cartesian product of \( Q \) and \( \mathcal{A} \) to \( Q \), and each total function \( \theta \) from \( Q \) to \( \mathcal{I} \),
if for each \( (q, \alpha) \) in the domain of \( \delta \),
\( \mathcal{F}(\theta(q), \alpha) \) is defined, and
\( \mathcal{F}(\theta(q), \alpha) \preceq \theta(\delta(q, \alpha)) \)
then \( \text{TEST}_1(\delta, \theta) = \text{‘true’} \)
otherwise \( \text{TEST}_1(\delta, \theta) = \text{‘false’} \), and
for some effective function \( \text{TEST}_2 \),
for each finite set \( Q \), each total function \( \theta_1 \) from \( Q \) to \( \mathcal{I} \) and each total function \( \theta_2 \) from \( Q \) to \( \mathcal{I} \),
if for each \( q \in Q \), \( \theta_1(q) \preceq \theta_2(q) \)
then \( \text{TEST}_2(\theta_1, \theta_2) = \text{‘true’} \)
otherwise \( \text{TEST}_2(\theta_1, \theta_2) = \text{‘false’} \).

Construct \( \text{RES} \) as follows:
for each feature structure \( (Q, q, \delta, \theta) \),
set \( \Sigma_{\text{in}} = \text{GEN}(Q) \) and \( \Sigma_{\text{out}} = \{ \} \)
while \( \Sigma_{\text{in}} \) is not empty
do set \( \Sigma_{\text{in}} = \{ \rho_1, \ldots, \rho_n \} \)
if \( \text{TEST}_1(\delta, \rho) = \text{‘true’} \),
\( \text{TEST}_2(\theta, \rho) = \text{‘true’} \), and
\( \Sigma_{\text{out}} = \{ \rho' \} \)
then set \( \Sigma_{\text{out}} = \{ \rho, \rho'_1, \ldots, \rho'_n \} \)
if \( \Sigma_{\text{out}} = \{ \rho_1, \ldots, \rho_n \} \)
then output \( \{ (Q, q, \delta, \theta_1), \ldots, (Q, q, \delta, \theta_n) \} \).
\( \text{RES} \) is an effective algorithm, and
for each feature structure \( F \),
\( \text{RES}(F) = \{ \text{the resolvants of } F \} \).

Theorem 27. If \( (\Omega, \mathcal{I}, \preceq, \mathcal{S}, \mathcal{A}, \mathcal{F}) \) is rational
and computable then for some effective function \( \text{SAT} \),
for each feature structure \( F \),
if \( F \) is satisfiable
then \( \text{SAT}(F) = \text{‘true’} \)
otherwise \( \text{SAT}(F) = \text{‘false’} \).

Proof. From theorem 24 and proposition 26.

Gerdemann and Götz’s T roll system (see [Götz 1993], [Gerdemann and King 1994] and [Gerdemann (fc)]) employs an efficient refinement of \( \text{RES} \) to test the satisfiability of feature structures. In fact, Troll represents each feature structure as a disjunction of the resolvants of the feature structure. Loosely speaking, the resolvants of a feature structure have the same underlying finite state automaton as the feature structure, and differ only in their output function. Troll exploits this property to represent each feature structure as a finite state automaton and a set of output functions. The Troll unifier is closed on these representations. Thus, though \( \text{RES} \) is computationally expensive, Troll uses \( \text{RES} \) only during compilation, never during run time.

References

[Carpenter 1992] Robert Carpenter The logic of typed feature structures. Cambridge tracts in theoretical computer science 32. Cambridge University Press, Cambridge, England. 1992.

[Gerdemann (fc)] Dale Gerdemann. Troll: type resolution system, user’s guide. Sonderforschungsbereich 340 technical report. Eberhard-Karls-Universität, Tübingen, Germany. Forthcoming.

[Gerdemann and King (1994)] Dale Gerdemann and Paul John King. The correct and efficient implementation of appropriateness specifications for typed feature structures. In these proceedings.

[Götz 1993] Thilo Götz. A normal form for typed feature structures. Master’s thesis. Eberhard-Karls-Universität, Tübingen, Germany. 1993.

[King 1989] Paul John King. A logical formalism for head-driven phrase structure grammar. Doctoral thesis. The University of Manchester, Manchester, England. 1989.

[Moore 1956] E. F. Moore. ‘Gedanken experiments on sequential machines’. In Automata Studies. Princeton University Press, Princeton, New Jersey, USA. 1956.
[Mosher 1988] Michael Andrew Mosher. *Extensions to unification grammar for the description of programming languages.* Doctoral thesis. The University of Michigan, Ann Arbor, Michigan, USA. 1988.