

Joint Utterance: Intrasentential Speaker/Hearer Switch as an Emergent Phenomenon

HASIDA, Kôiti

Electrotechnical Laboratory
1-1-4 Umezono, Tukuba
Ibaraki 305, Japan
hasida@etl.go.jp

NAGAO, Katashi

Sony Computer Science Lab. Inc.
3-14-13 Higashi-gotanda,
Shinagawa-ku, Tokyo 141, Japan
nagao@csl.sony.co.jp

MIYATA, Takashi

University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo 113, Japan
mya-u@is.s.u-tokyo.ac.jp

Abstract

A totally constraint-based computational architecture is proposed and applied to natural language processing. Here *constraint* means a design orthogonal to information flow, so as to implement diverse patterns of computation requisite in intelligent systems. In this architecture, declarative semantics is given by measuring the degree of violation of a constraint in terms of real valued *potential energy*, and context-sensitive control of computation emerges from the *dynamics* defined thereof.

Joint utterance with role switching between a speaker and a hearer in the midst of one sentence is accounted for as a corollary of constraint-based design plus this dynamical control, without recourse to any procedures specific to sentence parsing, generation, role switching, and so on. Since constraint does not refer to information flow, input and output are tightly coupled within a constraint module which is embodied as feedback loops coupling the system and its environment. Joint utterance emerges when the dynamical effect towards that constraint changes and flips this constraint between input and output, while some other constraints are continuously satisfied.

1 Introduction

The traditional design of AI systems based on procedural programming does not look feasible, because it stipulates information flow so that the complexity of the design heavily reflects the diversity of information flow requisite for intelligent behaviors. This diversity is very likely to be large enough to render such a system too complex and non-modular to extend or maintain.

This paper proposes a totally constraint-based computational architecture and applies it to natural language processing. Here *constraint* means a design which is orthogonal to information flow, so as to capture diverse patterns of computation in intelligent systems. This notion of constraint is very different from the ordinary one under which a constraint is a declarative knowledge to narrow down the search space. In our architecture, declarative semantics is defined by measuring the degree of violation of a constraint in terms of real valued *potential energy*, and context-sensitive control of computation emerges from the *dynamics* defined thereof.

Our concern here is what kind of mechanism might underlie joint utterance involving role switching between a speaker and hearer in the midst of one sentence. For in-

stance, it might be the case that one person opens up a sentence by saying *Kim* and another person succeeds the utterance and completes the sentence by saying *sneezed*. A standard way to explain such phenomena would be to postulate a procedure for sentence comprehension and another for sentence production, and finally to account for how to switch back and forth between them. One will then end up with a very cumbersome theory about how the two procedures could communicate, if they are different from each other as is the case with the procedural approach.

The constraint-based approach addresses quite another explanation. First of all, no specific procedures are postulated for the relevant tasks such as sentence parsing, generation, plan recognition, role switching, and so forth, since constraint is orthogonal to information flow. Moreover, joint utterance is accounted for as a corollary of constraint-based architecture. Since constraint does not refer to information flow, input and output are tightly coupled within a constraint module which is embodied as a feedback loop coupling the system and its environment. Joint utterance emerges when the dynamical field around that constraint changes and flips it from input and output or vice versa, the computation on the other relevant constraints such as syntax remaining on the previous tracks.

2 Computational Architecture

A quick overview of our computational framework is in order here. The subsequent sections contain further details just enough to account for the examples. See Hasida [1992] for a more comprehensive and general discussion.

We consider a clausal-form first order logic program, and treat it totally as constraint, in the sense that there is no predetermined direction of information flow such as top-down or left-to-right. Computation is controlled by a *dynamics* in the sense of physics. Dynamical interaction across the whole constraint implements omnidirectional information flow among various types of information, syntactic, semantic, pragmatic and so on.

2.1 Constraint Network

A constraint is a set of *clauses*. A clause is basically a disjunction of *literals*. A *literal* is an *atomic constraint* preceded by a sign. An atomic constraint is an *atomic formula* such as $p(X,Y,Z)$, a *binding* such as $X=f(Y)$, a *feature specification* such as $\text{agent}(X,Y)$, or an *equation* such as $X=Y$. A sign is '+', '-', or nil. For any atomic

constraint α , literal $+\alpha$ stands for just α , and $-\alpha$ stands for $\neg\alpha$. Names beginning with capital letters represent variables, and the others predicates. A clause is written as a sequence of the included literals followed by a period. The order among literals is not significant. So (1) and (2) represent the same clause, which means (3) in a rough, crisp approximation.

- (1) $\neg p(U,Y) + q(Z) - U=f(X) - X=Z.$
- (2) $+q(Z) - p(f(Z),Y).$
- (3) $\forall U, X, Y \{ \neg p(U,Y) \vee q(X) \vee U \neq f(X) \}$

A literal with a null sign is called the *head* of the clause; A clause with a head is called a *definition clause* of the predicate of the head. In a crisp approximation, the meaning of this predicate is defined in terms of a necessary and sufficient condition tailored from its definition clauses. For instance, if predicate p has the definition clauses in (4), then p is roughly defined by (5).

- (4) $p(X) - q(X,a). \quad p(f(X)) - r(X).$
- (5) $\forall A \{ p(A) \Leftrightarrow \{ \exists Y (q(A,Y) \wedge Y = a) \vee \exists Z (A = f(Z) \wedge r(Z)) \} \}$

Predicates having definition clauses are called *defined predicates*, and the other predicates *free predicates*.

A definition clause of **true** is called a *top clause*. A top clause corresponds to the query in Prolog. That is, top clause (6) represents top-level hypothesis (7).

- (6) **true** $\neg p(X) + q(X,Y).$
- (7) $\exists X, Y \{ p(X) \wedge \neg q(X,Y) \}$

Clause **+true**. forces the head **true** of some top clause to be true, so that the negation of the body of that top clause is hypothesized. The computation is to tailor the best hypothesis to explain the top-level one, in the spirit of Hobbs et al. [1990]. The top clause transforms as computation proceeds, when the information boundary between the system and its environment changes, or in particular when interactions with the external world take place.

A constraint is regarded as a network. For instance, the constraint in Figure 1 may be graphically repre-

- (a) **true** $\neg s(SEM, W0, W1) - speech(W0).$
- (b) $s(VSEM, X, Z) - agent(VSEM, NSEM)$
 $- np(NSEM, X, Y) - vp(VSEM, Y, Z).$
- (c) $np(kim, A, B) - "Kim"(A, B).$
- (d) $vp(SNEEZE, C, D) - sneeze(SNEEZE)$
 $- "sneezed"(C, D).$
- (e) $vp(VSEM1, E, G) - patient(VSEM1, NSEM1)$
 $- v(VSEM1, E, F) - np(NSEM1, F, G).$

Figure 1: A Constraint in Symbolic Form

sented as in Figure 2.¹ In such a graphical representation, a clause is a closed domain containing the atomic constraints constituting that clause. Atomic constraints without such indication are referred to as negative literals in clauses. An argument of an atomic constraint is depicted as a '•'. A term appearing as several arguments of atomic constraints in a clause is regarded as a hyperlink whose endpoints are those arguments. For instance,

¹Tense is neglected in the semantics of *sneeze* here.

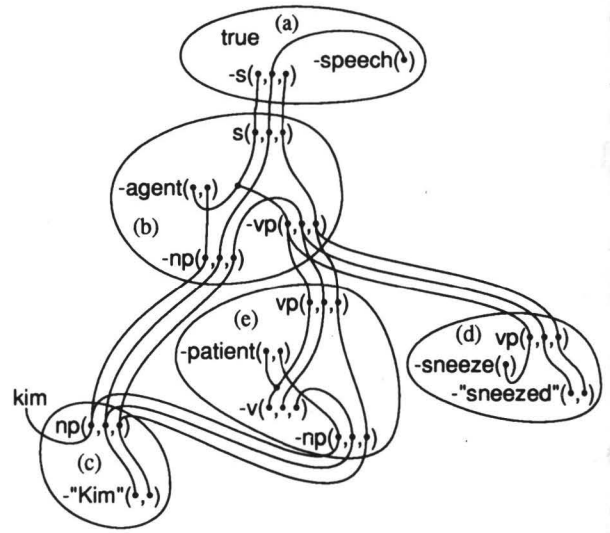


Figure 2: Constraint Network

term (variable) **VSEM** in clause (b) is a hyperlink with three endpoints. Constants such as **kim** in Figure 2 are also hyperlinks because they are terms. Only constants are hyperlinks running outside of clauses. Equations are links between arguments. For simplicity, we assume that no equation belongs to any clause.

We regard each part of constraint network as a set of instances, and consider that the unifiability of α and β implies that $\alpha \cap \beta = \emptyset$ is not known, for terms or atomic constraints α and β . We say α *subsumes* β to mean $\alpha \supseteq \beta$. We assume two atomic constraints are unifiable if and only if their corresponding arguments are directly connected through an equation, and that every equation connects two corresponding arguments of two unifiable atomic constraints. No equations are unifiable with each other.

We assume that initially all the atomic constraints with the same predicate are unifiable with each other, except that head atomic constraints are not unifiable with each other and non-head atomic constraints sharing the same defined predicate are not unifiable with each other. For example, the $vp(\bullet, \bullet, \bullet)$ s in (d) and (e) are not unifiable. Neither are the $np(\bullet, \bullet, \bullet)$ s in (b) and (e).

Not every equation is made explicit in a graphical representation. Normally only *strong* equations are displayed, but not *weak* ones, unless stated otherwise. Here a strong (weak) equation is one which supports a strong (weak) dynamical interaction between the two atomic constraints it links up. At the outset, the equations between atomic constraints with defined predicates are strong and the other equations are weak.

2.2 Dynamics and Computation

Each atomic constraint α has an *activation value* x_α , which is a real number such that $0 < x_\alpha < 1$ and may be regarded as the truth value (or a subjective probability of the truth) of α , zero and one corresponding to absolute falsity and absolute truth, respectively. The activation values of literals are defined so that $x_{+\alpha} = x_\alpha$ and $x_{-\alpha} = (1 - x_\alpha)$ for any atomic constraint α . The *potential energy* U of the entire constraint network is a function of the activation values, and represents the de-

gree of violation of the constraint. U gives rise to a *field of force* to change the state of the constraint network so as to decrease U .

Suppose there are n distinct atomic constraints in the given constraint, and hence n activation values, x_1 through x_n . Then the current state of the system is regarded as a point (8) in the n -dimensional Euclidean space, and the global potential energy U defines a field of force (9).

$$(8) \quad \vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad (9) \quad \vec{F} = \begin{pmatrix} -\frac{\partial U}{\partial x_1} \\ \vdots \\ -\frac{\partial U}{\partial x_n} \end{pmatrix}$$

$\vec{F}$ causes *spreading activation*. When $\vec{F} \neq \vec{0}$, a change of x_i so as to reduce U influences the neighboring parts of the constraint network, which causes further changes of activation values there, and thus state transition propagates across the network. In the long run, the assignment of the activation values will settle upon a stable equilibrium with $\vec{F} = \vec{0}$. The resulting state gives a minimal value of U . That is, the resultant $\vec{x}$ suboptimally satisfies the entire constraint.

The declarative semantics of the entire constraint is decomposed into several aspects. U is the sum of the local energies each representing one such aspect, so that U captures the global declarative semantics. The types of energy are *disjunction energy*, *exclusion energy*, *completion energy*, *assimilation energy*, *dependency energy*, *transitivity energy*, *binding energy*, and *normalization energy*. We discuss a few of them just below and some more afterwards.

The *disjunction energy* of a clause captures the disjunction of the literals: at least one literal should be true in a clause. Consider the following clause.

$$(10) \quad \neg p + q.$$

The disjunctive meaning of this clause is that either $\neg p$ or $+q$ should be true. The below disjunction energy represents the degree of violation of this meaning.

$$(11) \quad Dx_p(1 - x_q)$$

D is a positive constant associated with clause (10). Note that (11) is small if and only if either x_p or $1 - x_q$ is small; keep in mind that the activation values are between 0 and 1. In dynamical terms, q is excited by p to the extent that p is excited, and p is inhibited by q to the extent that q is inhibited. In Figure 2, $vp(\bullet, \bullet, \bullet)$ is excited by *sneeze*($\bullet$) and "sneezed"($\bullet, \bullet$) together in (d), due to the disjunction energy.

Exclusion energy represents the mutual exclusion of the literals in a clause, by which we mean that at most one literal should be true. So in (10), for example, only one of $\neg p$ and $+q$ may be true. This supports abductive inferences, to assume p when given q , and assume $\neg q$ when given $\neg p$. The exclusion energy of (10) is the following formula.

$$(12) \quad E(1 - x_p)x_q$$

E is a constant associated with clause (10). In Figure 2, *true* excites both *s*($\bullet, \bullet, \bullet$) and *speech*($\bullet$) due to the exclusion energy of the top clause, for example.

The *assimilation energy* between two unifiable atomic constraints captures the axiom of equality in a relaxed fashion: the activation values of two unifiable atomic constraints should be both close to zero or both close

to one to the extent that the equations between them are excited. So for instance $p(X, Y)$ and $p(U, V)$ tend to be both excited or both inhibited when the equation between X and U and that between Y and V are both strongly excited. Then the assimilation energy of the pair of two unifiable atomic constraints α and β is defined by the following.

$$(13) \quad -A_\pi s_{\alpha\beta} \left(x_\alpha - \frac{1}{2}\right) \left(x_\beta - \frac{1}{2}\right) \prod_\delta x_\delta$$

A_π is a positive constant associated with predicate π shared by α and β . δ ranges over the equations connecting the corresponding arguments of α and β . $s_{\alpha\beta}$ is a non-negative constant called the *subsumption coefficient* between α and β . When α and β are unifiable, $s_{\alpha\beta} = 1$ if α is known to subsume β or vice versa, and otherwise $s_{\alpha\beta} = s_0$ for a constant s_0 such that $0 < s_0 < 1$. So α and β strongly influence each other when they are in subsumption relation. When α and β are not unifiable, $s_{\alpha\beta} = 0$.

A *transitive cycle* is a cycle of equations in which any two adjacent equations are connected with different arguments filled by the same term. The *transitivity energy* of a transitive cycle Δ is defined as follows, which captures the axiom of equality, excluding the cases where just one equation in Δ is false.

$$(14) \quad \begin{cases} -t \prod_\delta (x_\delta - \theta) & (s_\delta x_\delta < \theta \text{ for at most one } \delta) \\ 0 & (\text{otherwise}) \end{cases}$$

δ ranges over the equations in Δ , and s_δ is again the subsumption coefficient of the two atomic constraints connected by δ . θ is a constant such that $0 < \theta < 1$. t is a positive constant. So the transitivity energy is large when just one δ has an activation value smaller than θ . Due to this energy, the equations in a transitive cycle very often tend to excite each other.

Symbolic operations encompass not only internal computation including *subsumption operation* (a generalization of unification to create subsumption relation instead of equality) and deletion (of links and clauses) but also input (perception) and output (action). Some of these operations are accounted for along the examples discussed in the subsequent sections. The preference for a symbolic operation depends on the expected degree to which the accompanying local change may dynamically influence some central part of the constraint. This degree is approximated by $|\frac{\partial P}{\partial s}|$, where penalty function P is the disjunction energy of clause $+true$. at the equilibrium of spreading activation, and s is some parameter of the potential energy which is altered by the operation in question. $|\frac{\partial P}{\partial s}|$ is regarded as the importance of parameter s . This value is computed by a generalized backpropagation.

Input and output are handled by constraints on the phenomena directly perceivable (in the sense of Gibson [1979]) in the environment, such as *speech*($\bullet$) in Figure 2. Such constraints extend over both the cognitive system and the world. Namely, they implement feedback loops connecting the system and the world, whereas the other constraints implement feedback loops within the system. Any constraint cannot be just for input or just for output, because input and output are ways of information flow and hence abstracted away from the level of constraint. Since information flow is determined according to dynamical context in our architecture, whether

speech(•) serves input or output also varies from one context to another. Generally speaking, such a perceptual constraint, when fired, introduces a novel piece of information about the external world. For instance, **speech**(•) unfolds the linear structure of speech. The difference between input and output is just that in the case of output the introduced information is attributed to the system's own action.

Here it is important that input and output are so tightly coupled for each sensorimotor feedback loop, because this is the source of intrasentential switch of speaker and hearer. That is, joint utterance is almost a corollary of constraint-based design, as exemplified in the subsequent sections.

This architecture is omni-directional and holistic for the following reasons. First, the spreading activation based on potential energy implements isotropic information flow, because any two activation values influence each other through dynamical interactions. Second, in this connection, backpropagation is also bidirectional in the same sense, the importance of any two parameters affecting each other to the extent that those parameters are related. Third, symbolic operations do not follow any prescribed order or direction, but controlled on the basis of such dynamical properties. Each symbolic operation is affected by the local dynamical context, which reflects the global context due to backpropagation.

3 From Comprehension to Production

In this section we discuss how the initial hearer may start speaking in accordance with the previous context. The scenario is that a sentence begins with an utterance of one word but the rest of the sentence is taken over by the initial hearer, when, for instance, the initial speaker cannot recall the right word to follow.

Figure 3 shows a part of the constraint when sentence

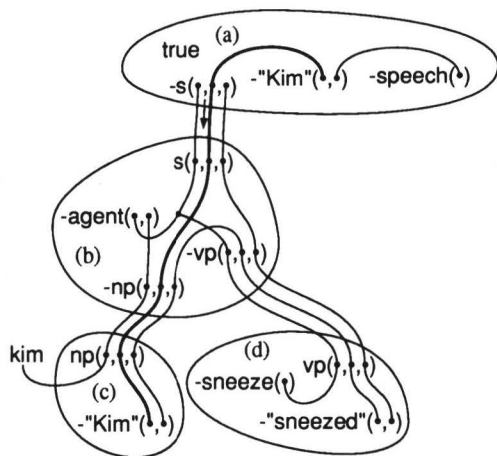


Figure 3: Starting Sentence Comprehension

comprehension is to start. The clauses are the same as in Figure 2 except that the top clause (a) has incorporated new literal **-Kim**(•,•), which corresponds to the observation of an utterance of word "Kim" in the external environment. Clause (e) is omitted because it does not play any important role in the rest of the story.

As indicated by the thick curve, the first arguments of the two **"Kim"**(•,•)s are connected through a *depend-*

ency path, by which we mean a path of strong equations which implies that the terms on both ends are unifiable. In general, when two terms are connected through a dependency path and fill the first arguments of two unifiable bindings or two unifiable feature specifications, subsumption operations are triggered so that one of those terms should subsume a copy of the other. Those subsumption operations start from the term which is in the more important binding or feature specification, and proceed along the dependency path. The importance of an atomic constraint α is measured by $|\frac{\partial P}{\partial s}|$. s is the parameter of dummy (in the sense that s is always zero energy sx_α , which corresponds to an external input to α). In general, a subsumption concerns a pair of unifiable atomic constraints, as is the case with unification, and the preference for a subsumption along a dependency path is given by $|\frac{\partial P}{\partial s}|$ again, where s is the subsumption coefficient of the pair of the relevant atomic constraints.

In the current example, the first argument of the upper **"Kim"**(•,•) is going to subsume that of the lower **"Kim"**(•,•), because the former **"Kim"**(•,•) is in the top clause and thus is more important. So the first subsumption is the one indicated by the arrow in Figure 3. This gives rise to the next state shown in Figure 4. Now clause

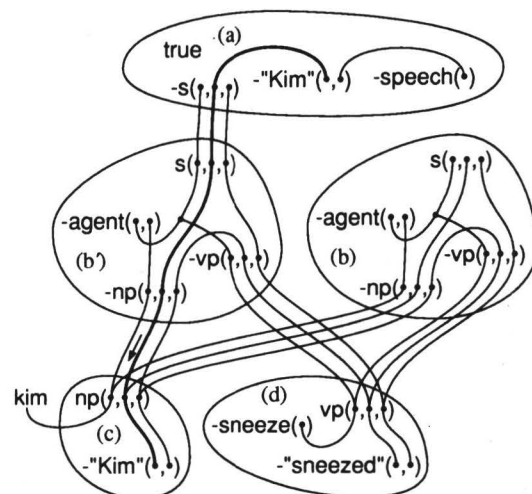


Figure 4: Recognizing the First Word

(b) in the previous state has been duplicated to be (b) and (b'). (b') is the copy whose $s(•,•,•)$ is the original $s(•,•,•)$'s maximum subset subsumed by the $s(•,•,•)$ in (a). So the $s(•,•,•)$ in (b) is not unifiable with that in (a) any more now.

The subsumption along the thick dependency path proceeds further down, as indicated by the arrow in Figure 4. The result is given in Figure 5. Here the first argument of the **"Kim"**(•,•) in (a) subsumes that of the **"Kim"**(•,•) in (c'). So the former **"Kim"**(•,•) subsumes the latter, because **"Kim"** is a partial function, in the sense that if **"Kim"**(α, β) and **"Kim"**(α, γ) are both true then β and γ must be identical. In general, equations between two atomic constraints are rendered strong when one of these atomic constraints subsumes the other. This is why the equations between the two **"Kim"**(•,•)s are shown Figure 5.

Now we consider how the initial hearer could take over

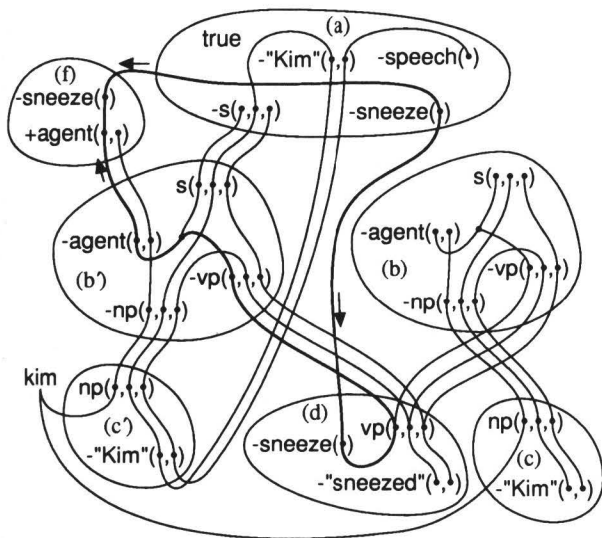


Figure 5: Switching to Production

with (a) and (b') and (d') is the instance of (d) linked with (a). Then the current configuration is as shown in Figure 6, where unimportant clauses are omitted.

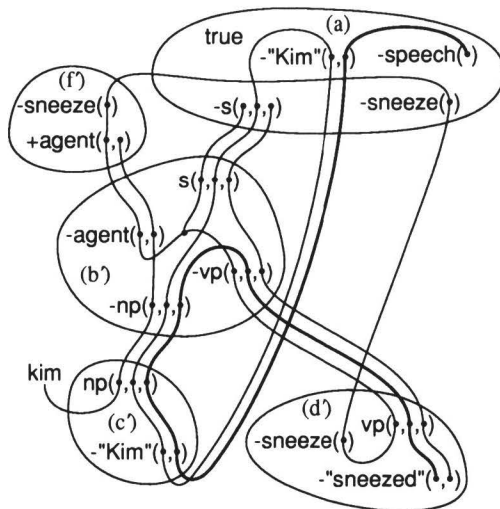


Figure 6: Triggering Utterance

for the rest of the sentence by saying *sneezed*. Let us investigate how that may happen under the assumption that the top clause (a) contains a literal **sneeze(•)**, which means that the initial hearer believes that somebody sneezed. She does not have to know from the outset that it is Kim that sneezed, because she can abductively guess it as shown later. Let us also postulate the following clause, which means that a sneezing event should have an agent.

(f) **-sneeze(U) +agent(U,V).**

Then the **sneeze(•)** in (a) may excite that in (f) due to the assimilation energy. That would excite the **agent(•,•)** in (f) due to the disjunction energy of (f), which will further excite **agent(•,•)** in (b') due to the assimilation energy again. On the contrary, the **agent(•,•)** in (b') excites **agent(•,•)** in (f), it excites **sneeze(•)** in (f), and it excites **sneeze(•)** in (a). This motivates the two indicated subsumptions from (a) and (b') to (f),² because that will more strongly excite the **sneeze(•)** in (a) and **agent(•,•)** in (b') and thus reduce the penalty function P . That is, their mutual activation through (the instance of) (f) will become stronger because the relevant links will be strong due to the subsumptions. This is predicted because $-\frac{\partial P}{\partial s}$ is large for the subsumption coefficient s between the two **sneeze(•)**s and also the two **agent(•,•)**s. The indicated subsumptions amount to an abductive inference to the effect that it is Kim that sneezed. Note that this process captures marker passing [Charniak, 1986; Norvig, 1989] as an emergent phenomenon of the dynamics, markers corresponding to high activation values. On the other hand, the subsumption operation from **sneeze(•)** in (a) to that in (d) could also happen, because that will decrease P , too. Note that the three subsumptions indicated in Figure 5 are all along the cycle of equations depicted as the thick closed curve. This cycle increases the preferences of these subsumptions, because the equations on it tend to have relatively high activation values due to the transitivity energy.

Now suppose (f') is the resulting instance of (f) linked

Here let us assume that **speech(•)** produces a word. This happens when the phonological form of a word connected with **speech(•)** through a dependency path is sufficiently important in the sense described before. All the perceptual constraints behave essentially the same way. In Figure 6, the phonological form "**sneezed**"(•,•) is connected via the thick dependency path with **speech(•)**, and also important because (d') is linked with (a) and hence this "**sneezed**"(•,•) has stronger interactions with the top clause than the other phonological word forms.

If **speech(•)** is thus fired and transformed while introducing a new piece of external information about the utterance of "sneezed," then the resulting structure of the relevant part of the constraint will be as shown in Figure 7. Now arises the thick dependency path, which

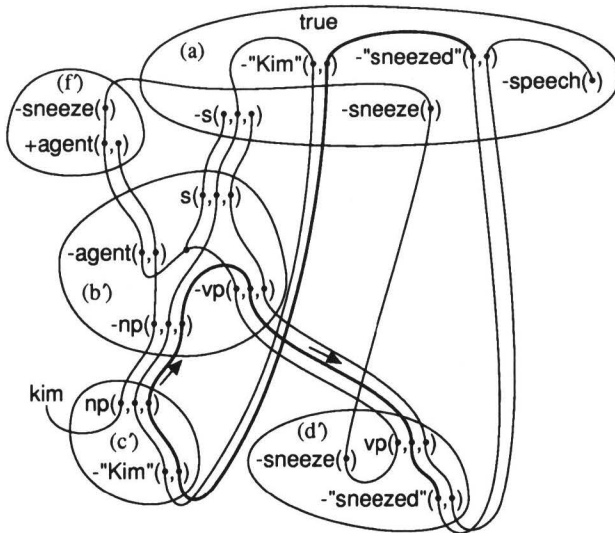


Figure 7: Sentence Finished

triggers the subsumption operations indicated by the ar-

²The lower subsumption might be in the opposite direction if it took place after the upper one.

rows. As a result, the copy of "sneezed"($\bullet, \bullet$) in an instance of (d') is subsumed by the "sneezed"($\bullet, \bullet$) in (a), so that the equations between the two "sneezed"($\bullet, \bullet$)s are rendered strong. So the sentence is done.

4 From Production to Comprehension

Here we consider the initial speaker, who becomes the hearer from midst of the sentence. Let us suppose that she begins with the constraint shown in Figure 8. Here

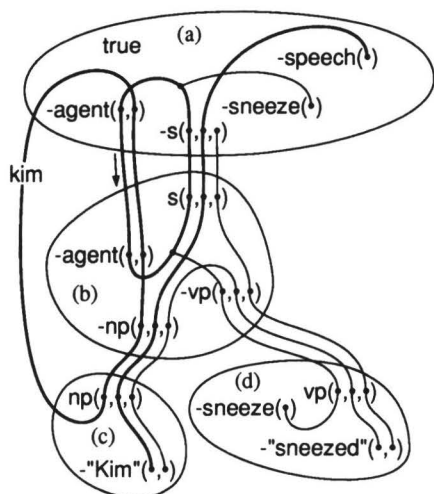


Figure 8: Starting Sentence Production

the semantic content to be verbalized is in the top clause. Two thick cycles of equations are shown here, each consisting of two external equations, one weak and the other strong. The weak equations are those between the two **agent**($\bullet, \bullet$)s. The subsumption operation indicated by the arrow is preferred here because that would increase the mutual excitation of the two **agent**($\bullet, \bullet$)s, which contributes to the reduction of the penalty function. Although not explicitly shown, another cycle of equations runs through the two **sneeze**($\bullet$)s in (a) and (d), and thus a subsumption operation between them is also motivated here. However, the rest of the discussion assumes the former subsumption.

Here the reader might want to compare this with semantic-head-driven generation (SHDG) [Shieber *et al.*, 1989]. The two subsumptions mentioned above both correspond to retrieval of the semantic head (the minimal grammatical category with the same semantics as the root category to be generated) in SHDG. Our method is more general and flexible, however, because its computation follows the emergent control derived from dynamics, whereas SHDG is a procedure prescribing information flow. For instance, the above subsumptions are not prescribed at all but motivated on a very general ground depending on the dynamical context.

As indicated by the thick dependency path between **speech**($\bullet$) and "Kim"($\bullet, \bullet$), the word "Kim"($\bullet, \bullet$) is now focused on as a promising candidate for the first word, because the equation between the first arguments of the two **np**($\bullet, \bullet, \bullet$)s is particularly excited due to the cycle of equations running through **kim**.

Let us assume that the subsumption takes place before the utterance of *Kim*. This amounts to hypoth-

esizing that the speaker is relatively deliberate rather than reactive, in the sense that internal inferences are preferred to actions. Of course we can encode different ratio of deliberation and reactivity in terms of dynamical parameters which control symbolic operations including actions, in a way similar to Maes [1991].

The subsumption in question results in Figure 10. Since the feature specification **agent**($\bullet, \bullet$) in (b') is the

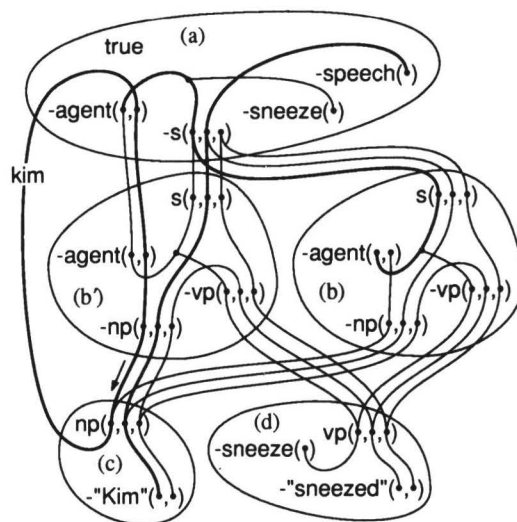


Figure 9: Tailoring the First Word

maximum subset of the original subsumed by **agent**($\bullet, \bullet$) in (a), **agent**($\bullet, \bullet$) in (b) is not unifiable with the other two. So the dependency between the first arguments of the two **agent**($\bullet, \bullet$)s in (a) and (b) indicates a contradiction, because if these arguments are identical then these feature specifications must unify since a feature is a partial function. In general, a local energy called the *dependency energy* is defined for each dependency path connecting two terms not unifiable. This energy engenders mutual inhibition among the atomic constraints on both ends of the dependency path and the equations in the path. In Figure 9, therefore, the two **agent**($\bullet, \bullet$)s in (a) and (b) and the equation connecting their first arguments inhibit each other. Here the **agent**($\bullet, \bullet$) in (a) will defeat the others because it is in the top clause and hence receives the strongest excitation. So the **agent**($\bullet, \bullet$) in (b) or the equation connecting the first argument of the two **s**($\bullet, \bullet, \bullet$)s is inhibited, which renders (b) unimportant in the rest of the current task. We will thus neglect (b) henceforth.

The subsumption indicated in Figure 9 is motivated because of the dependency path connecting **kim** with itself. After this subsumption, we arrive at the situation in Figure 10. Now let us suppose that for some reason or another the hearer happens to say *sneezed* before the speaker does. After the input of *sneezed*, the top clause (a) incorporates a new atomic formula "sneezed"($\bullet, \bullet$). Now the first arguments of the two "sneezed"($\bullet, \bullet$)s are connected through a dependency path, which is the same as the one indicated in Figure 7. So subsumptions are triggered also as shown in Figure 7, and the sentence is completed.

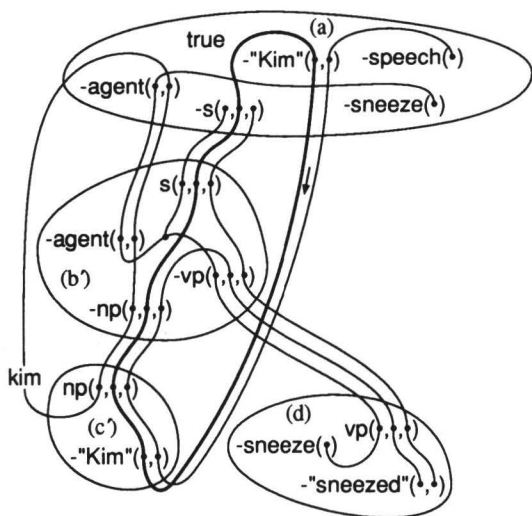


Figure 10: Switching to Comprehension

5 Concluding Remarks

Joint utterance has been accounted for as a corollary of constraint-based design. The essence of the account is that input and output are tightly coupled as parts of feedback loops constituting a constraint. Transfer of conversational roles in the middle of a sentence hence naturally emerges without recourse to any special mechanism. It is simply that the dynamical context around that constraint flips it between input and output. The reported work has been fully implemented. The program is written in C and runs in UNIX with X window interface.³

The present work is related to several other studies in such a way that it integrates and generalizes some aspects of them. First, it is based on the work on reversible natural language processing [Hasida, 1990; Shieber, 1988]. Since our framework is constraint-based, it should fit in the picture of reversible NLP in a strongest sense that sentence comprehension and production are based on the same linguistic and extralinguistic constraints and the same general procedure to deal with constraints. The current study is an extension of Hasida [1991], which is the only work along this strongest sense. Second, our work may be compared with connectionist NLP [Kalita and Shastri, 1987; Waltz and Pollack, 1985]. In this respect, our approach is distinguished in that it captures declarative semantics of logic programs in dynamics and derives control of computation thereof. In the connectionist approaches, either network structure is fixed or control of symbolic computation is programmed more arbitrarily and less systematically. Finally, as mentioned before, the above example on intrasentential role switching has a lot in common with reactive planning [Maes, 1991] and time-dependent planning [Boddy and Dean, 1989]. The unique characteristic of our work in this connection is the integration of declarative semantics and computation control again, and practically full first-order descriptive capacity.

Future work includes applications to wider range of problems and extension of the computational architec-

ture. An ultimate goal into the former direction is to implement a full NLP system in the constraint-based architecture as a huge bunch of feedback loops running both the inside and outside of the system. Regarding the latter direction, we want to tailor a learning algorithm for at least parameters of the potential energy. Also, it is definitely necessary to implement the architecture on massively parallel hardware.

References

- [Boddy and Dean, 1989] M. Boddy and T. Dean. Solving time-dependent planning problems. In *Proceedings of IJCAI '89*, pages 979–984, 1989.
- [Charniak, 1986] E. Charniak. A neat theory of marker passing. In *Proceedings of AAAI '86*, pages 584–588, 1986.
- [Gibson, 1979] J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin Company, 1979.
- [Hasida, 1990] K. Hasida. Sentence processing as constraint transformation. In *Proceedings of ECAI '90*, pages 339–344, Stockholm, 1990. (An extended version in *Proceedings of LPC'90*, Tokyo.).
- [Hasida, 1991] K. Hasida. Common heuristics for parsing, generation, and whatever ... In *Proceedings of the Workshop on Reversible Grammar in Natural Language Processing*, pages 81–90, Berkeley, 1991. held in connection with 29th Annual Meeting of the Association for Computational Linguistics.
- [Hasida, 1992] K. Hasida. Dynamics of symbol systems: An integrated architecture of cognition. In *Proceedings of FGCS '92*, Tokyo, 1992.
- [Hobbs et al., 1990] J. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. Technical Note 499, SRI International, 1990.
- [Kalita and Shastri, 1987] J. Kalita and L. Shastri. Generation of simple sentences in English using the connectionist model of computation. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*. Laurence Erlbaum, 1987.
- [Maes, 1991] P. Maes. Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 49–70. MIT/Elsevier, 1991.
- [Norvig, 1989] P. Norvig. Marker passing as a weak method for text inferencing. *Cognitive Science*, 13:569–620, 1989.
- [Shieber et al., 1989] S. M. Shieber, G. van Noord, and R. C. Moore. A semantic-head-driven generation algorithm for unification-based formalisms. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 7–17, 1989.
- [Shieber, 1988] S. M. Shieber. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 614–619, 1988.
- [Waltz and Pollack, 1985] D. Waltz and J. Pollack. Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science*, 9:51–74, 1985.

³Its source code is available as a part of ICOT Free Software. Please contact the first author for the newest version.