1 はじめに

実用的な対話処理システムには, 文法的に不適格な文に対しても頑健に対処できることが要求されるが, これまでに作成されたシステムの多くは, 文法や辞書によって定められた適格な文を解析の対象としていた. これに対して, 文法的に不適格な文を頑健に処理する枠組がこれまでにいくつか提案されている. そのような枠組として, 単一化文法を拡張した手法 [3, 5], 構文解析アルゴリズムを拡張した手法 [4, 8, 11, 15, 18], 部分的に作成された構文構造から情報を抽出する部分解析法 [7, 12], 言い直しや助詞の省略といった特定の不適格な現象を処理する手法 [2, 16, 17, 19] などがある.

ところで, 実時間対話処理システムを実現するためには, 自然言語文をその出現順序に従って順次解釈する枠組, すなわち, 漸進的解釈手法が不可欠となる [6]. 相手の発話を漸進的に解釈することにより, 発話途中での割り込みの生成や即時的な応答が可能となり, 自然で円滑な対話の実現が期待できる. さらに, 漸進的かつ頑健な意味解釈を実現するためには, 文法的不適格文に対して, 入力途中の段階でそれまでの入力に対する構文構造を作成し, これを解釈する枠組が必要である.

そこで本稿では, 文法的不適格文を漸進的に構文解析するための手法を提案する. すなわち, 本手法では, 適格文に対する漸進的な構文解析を実現する漸進的チャート解析 [13] に, 誤り修正の手続きを導入する. また, 本手法では, 到達可能性, 及び連接可能性といった構文的制約を用いて, 文の入力途中の段階で随時, 誤り修正を実行する. これにより, 文法的不適格文に対して漸進性を損なうことなく構文構造を作成することができる. さらに, 誤り修正に対してコストを導入することにより, 入力文に対する修正の度合を表現する. 文法的不適格文も部分的には適格であり, その適格な部分に対する構文構造が存在すると考えられるが, 構文構造にコストを付与することにより, 適格な部分に対する構造をより多く保存している構文構造の選択が可能となる.

本稿の構成は以下の通りである. 次の 2 節では, 不適格文解析手法としての本手法の位置づけを行う. 3 節では, 漸進的チャート解析について説明する. 4 節では, 文法的不適格文に対する漸進的解析における誤り修正手法について述べ, 5 節では, その例を示す. 最後に 6 節で, 関連研究との比較を行う.

2 従来の不適格文解析手法

文法的不適格文を解析する手法は, これまでにいくつか提案されている. 本節では, これらについて概観し, 本稿で提案する手法の位置づけを行う.

単一化文法に基づく不適格文解析手法として, Douglas ら [3] は PATR に基づく手法を, 今一ら [5] は HPSG に基づく手法を提案している. これらは, 性や数の一致などといった文の構成要素間の制約に関する不適格性を扱う. Douglas らの手法は, 入力文の解析に失敗すると, この制約を段階的に緩和する. また, 今一らの手法は, 制約を必ずしも満足する必要はないものとして捉え, より多くの制約を満足する構造を解析結果とする.

構文解析アルゴリズムに基づく不適格文解析手法として, Mellish[15] や加藤 [8] はチャート解析に基づく手法を, 斎藤 [18] や今井ら [4] は GLR 法に基づく手法を, Lee ら [11] はアーリー法に基づく手法をそれぞれ提案している. これらの手法はいずれも, 語の脱落, 挿入, 置換といった構文的な不適格性を含む入力文に対して誤り修正を行うことにより, 解析を実行する.

Jensen ら [7] や McDonald[12] は, 文法的不適格文に対して, 文全体に対する構文構造を作成するのではなく, 断片的な構文構造を作成し, それらの構文構造から意味を抽出する手法を提案している. これらは部分解析法と呼ばれる.

また, 不適格な現象を個別に扱う手法として, 言い直しを扱う手法 [2, 16, 17] や, 助詞の省略を扱う手法 [19] が提案されている.

本稿で提案する手法は, 文献 [4, 8, 11, 15, 18] で扱われている構文的な不適格性を対象とし, 構成要素間の制約に関する不適格性は対象としない. また, 文献 [2, 16, 17, 19] と同様に, 不適格な現象に関する知識を用いることにより, より効果的な解析の実現が期待できるが, これについては本稿では論じない.

3 漸進的チャート解析

本節では, 漸進的チャート解析 [13] について簡単に説明する. この手法は, 単語が入力されるごとにそれまでの入力に対する構文構造を随時作成するため, 入力途中の段階で構文構造を用いた処理, 例えば, 意味解析や文脈解析, あるいは翻訳処理などを実行するための枠組として適している.

漸進的チャート解析は従来のチャート解析 [10] と同様に, チャート (chart) と呼ばれるグラフ構造を用いて解析結果を保持する. チャートは節点の集合, 及び弧の集合から構成される. 節点 (node) は入力文中の語と語の間に位置する. i 番目の語 w_i と $i+1$ 番目の語 w_{i+1} の間の節点は, 番号 i でラベル付けされる. 以下では, 番号 i でラベル付けされた節点を単に節点 i と呼ぶ. 弧 (edge) は節点と節点を結び, その弧が覆っている部分に対する構文構造をラベルとしてもつ. この構文構造を項 (term) と呼ぶ. 項のうち, 未入力のためその構造が決定されていないものを未決定項 (undecided term) と呼び, 記法 $[?]_X$ で表現する. 弧にラベルづけされた項が未決定項を含むとき, この弧を活性弧 (active edge) と呼び, そうでないとき, 不活性弧 (inactive edge) と呼ぶ. 活性弧の項のうち, もっとも左に位置する未決定項を最左未決定項 (leftmost undecided term) と呼ぶ.

漸進的チャート解析では, 活性弧の最左未決定項を別の項で置き換えることにより解析が進行する. i 番目の語 w_i が入力された時点で実行される手続きは以下の 3 つである.

文法		辞書	
s	→ np vp	pron	→ I
np	→ pron	det	→ the
np	→ det n	n	→ train
np	→ gi pp	gi	→ going
np	→ n	p	→ by
vp	→ vt s	vt	→ think
vp	→ vi pp	vi	→ think
vp	→ be adj	be	→ is
pp	→ p np	adj	→ best

s : 文	np : 名詞句	vp :動詞句
pp :前置詞句		
pron: 代名詞	det: 冠詞	n : 名詞
vt : 他動詞	vi : 自動詞	gi :動名詞
p : 前置詞	be :be 動詞	adj:形容詞

図 1: 構文解析のための文法と辞書

なお, 以下では A, X, Y, Z は範疇を表すものとする.

辞書引き 語 w_i の範疇が X ならば, 項 $[w_i]_X$ をラベルとしてもつ不活性弧をチャートの節点 $i-1$ と i の間に追加する.

文法規則の適用 チャートの節点 $i-1$ と i を結ぶ項 $[\cdots]_X$ をラベルとしてもつ弧に対して, 文法規則 $A \rightarrow XY \cdots Z$ が存在するならば, 項 $[[\cdots]_X[?]_Y \cdots [?]_Z]_A$ をラベルとしてもつ弧をチャートの節点 $i-1$ と i の間に追加する.

項の置き換え チャートの節点 0 と $i-1$ を結ぶ活性弧の項 σ の最左未決定項を $[?]_X$ とする. このとき, チャートの節点 $i-1$ と i を結ぶ弧の項 τ の範疇が X ならば, 項 σ の最左未決定項を項 τ で置き換えた項をラベルとしてもつ弧をチャートの節点 0 と i の間に追加する.

漸進的チャート解析は, 従来の上昇型チャート解析と異なり, 活性弧に対して文法規則を適用する操作, 及び活性弧の項の最左未決定項を別の活性弧の項で置き換える操作を導入している. これにより, 入力途中の段階でそれまでの入力に対する構文構造を漸進的に作成することが可能となる.

例えば, 図 1に示す文法と辞書を用いるとき, 英語文

(3.1) I think going by train is best.

における語 "think" が入力された段階までの解析処理を比較すると, 通常の上昇型チャート解析では, "I", 及び"think" にそれぞれ文法規則を適用することにより, 構文構造 $[[[I]_{pron}]_{np}[?]_{vp}]_s$, $[[think]_{vi}[?]_{pp}]_{vp}$, $[[think]_{vt}[?]_s]_{vp}$ が作成されるが, 漸進的なチャート解析では "I think" に対する構文構造

(3.2) $[[[I]_{pron}]_{np}[[think]_{vt}[?]_s]_{vp}]_s$
(3.3) $[[[I]_{pron}]_{np}[[think]_{vi}[?]_{pp}]_{vp}]_s$

をそれぞれ作成することができる. このような構文構造を作成することにより, 入力途中の段階で文の構成要素間の関係, 例えば, "think" の主語は "I" であるといった関係を捉えることができるため, 早い段階での意味解析や文脈解析が可能となる [1].

4 文法的不適格文に対する漸進的構文解析手法

本稿では, 構文的な不適格性を入力文中の誤りとして捉える. すなわち, 本稿で提案する文法的不適格文に対する漸進的解析手法は, 漸進的チャート解析に対して誤り修正の手続きを導入した枠組である. 文法的不適格文に対して, 語が入力されるごとに随時, 誤り修正を実行し, それまでの入力に対する構文構造を作成する. 本節では, 誤り修正手法について述べる.

4.1 誤りの種類とその修正

本手法で扱う構文的な誤りは, 文献 [4, 8, 11, 15, 18] と同様, 必要な語の脱落 (脱落誤り), 余分な語の挿入 (挿入誤り), 別の語への置換 (置換誤り) の 3 種類とし, 入力文中にこれらの誤りが複数出現することを許す. ただし, 誤りは連続して出現しないものとする.

これらの誤りはそれぞれ,

- 語を挿入する (脱落誤り修正)
- 語を読み飛ばす (挿入誤り修正)
- 別の語に置き換える (置換誤り修正)

ことにより修正することができる. 例えば, 文法的に不適格な英語文

(4.1)* I think by train is best.

は, これら 3 種類の誤りに応じて, (1)"think" と "by" の間から範疇が動名詞である語が脱落した文, (2) 語 "by" が余分に挿入された文, (3) 範疇が冠詞である語が語 "by" で置き換えられた文, であると考えることができる.

(1) の場合, 範疇が動名詞である語, 例えば "going" を挿入することによって誤りを修正できる. 実際, 語 "think" の直後に位置する節点を始点かつ終点とし, 項 $[going]_{gi}$ をラベルとしてもつ弧をチャートに追加し, これに対して文法規則を適用すると, 項

(4.2) $[[[going]_{gi}[?]_{pp}]_{np}[?]_{vp}]_s$

をラベルとしてもつ弧を作成できる. さらに, 項 (3.2) の最左未決定項 $[?]_s$ を項 (4.2) で置き換えることにより, "I think going" に対する構文構造

(4.3) $[[[I]_{pron}]_{np}[[think]_{vt}[[[going]_{gi}[?]_{pp}]_{np}[?]_{vp}]_s]_{vp}]_s$

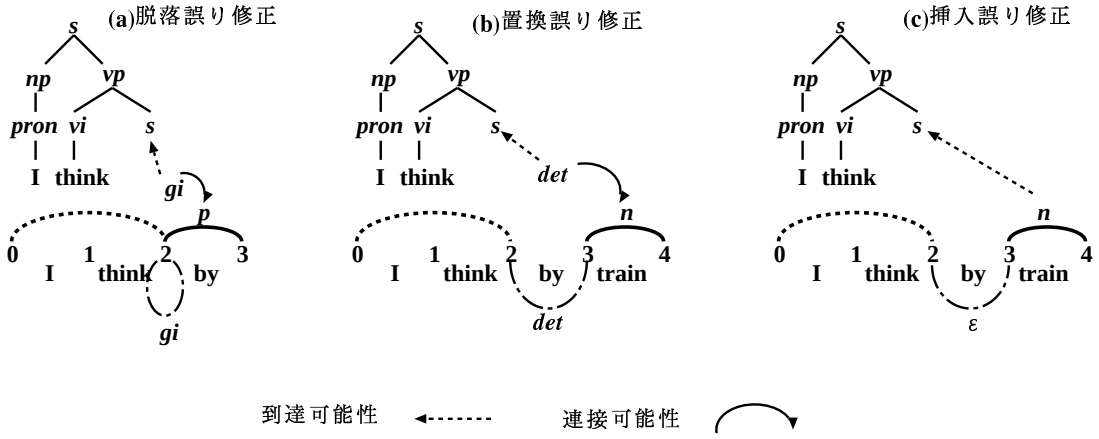

図 2: 誤り修正の例

を作成できる.

また, (2) の場合, 語 “by” を読み飛ばすことにより誤りを修正できる. これにより, “I think train” に対する構文構造

(4.4) $[[[I]_{pron}]_{np}[[think]_{vt}[[[train]_{n}]_{np}[?]_{vp}]_{s}]_{vp}]_{s}$

を作成できる.

さらに, (3) の場合, 語 “by” を範疇が冠詞である語, 例えば “the” に置き換えることによって誤りを修正できる. すなわち, “I think the” に対する構文構造

(4.5) $[[[I]_{pron}]_{np}[[think]_{vt}[[[the]_{det}[?]_{n}]_{np}[?]_{vp}]_{s}]_{vp}]_{s}$

を作成できる.

以上のように, 誤り修正を行うことにより, 入力途中の段階で随時, 文法的不適格文に対して構文構造を作成できる.

4.2 誤りの種類の絞り込み

前節では, 誤り修正を実行することにより, 文法的不適格文に対する構文構造を作成した例を示した. しかし, 上述した 3 種類の修正を適切に実行しなければ構文構造を作成することはできない. また, 解析効率の低下を防ぐために, 構文構造の作成に寄与しない不要な誤り修正の実行を回避しなければならない. すなわち, 可能な誤り修正の種類を前もって絞り込み, 効率的に構文構造を作成する必要がある. さらに, 漸進的な解析を実現するために, 入力途中の早い段階でこのような絞り込みを実行することが望まれる. そこで本手法では, これまでの解析結果, 及び先読みした語のもつ情報を用いて誤りの種類を絞り込み, 誤り修正を随時実行することにより, 漸進的に構文構造を作成する.

4.2.1 到達可能性・連接可能性

まず, 誤りの種類を絞り込むときに用いる到達可能性, 及び連接可能性について説明する. これらはいずれも範疇間の関係を表す. なお, 以下では A, X, Y, Z を範疇とし, α, β を範疇の列とする.

まず, 到達可能性の定義を示す.

定義 **4.1**(到達可能性) 範疇 X が範疇 Y に到達可能 (reachable) であるとは, X, Y が次の条件 1. ～ 3. のいずれかを満たすときである.

1. $X = Y$.
2. 文法規則 $Y \rightarrow X\alpha$ が存在する.
3. X が Z に到達可能で, かつ Z が Y に到達可能であるような範疇 Z が存在する. □

X が Y に到達可能であるとは, Y を根とし, X をその左端の子とする構文構造が存在することを意味する. 以下では, 記法 $X \rightsquigarrow Y$ で範疇 X が範疇 Y に到達可能であることを表す.

次に, 連接可能性の定義を示す.

定義 **4.2**(連接可能性) 範疇 X が範疇 Y に連接可能 (connectable) であるとは, X, Y が次の条件 1., 2. のいずれかを満たすときである.

1. 文法規則 $A \rightarrow \alpha X Z \beta$ が存在し, かつ $Y \rightsquigarrow Z$.
2. 文法規則 $A \rightarrow \alpha X$ が存在し, かつ A が Y に連接可能である. □

範疇 X が範疇 Y に連接可能であるとは, X が Y の左隣に位置できることを意味する. 以下では, 記法 $X \frown Y$ で範疇 X が範疇 Y に連接可能であることを表す.

4.2.2 誤り修正の手続き

本手法では, 語が入力されるごとに誤り修正を随時実行する. i 番目の語 w_i が入力されたときに行う誤り修正の手続きを以下に示す. なお本手法では, 文法規則の情報を用

いて誤りを絞り込み, 誤り修正により挿入, あるいは置換された語の範疇を求めるが, その語が何であるかを具体的に同定することはできないため, 特殊な語 $*$ を用いてこれを表現する. また, X を範疇, σ を項とするとき, 記法 $X \rightsquigarrow \sigma$ で X が σ の最左未決定項の範疇に到達可能であることを表し, 記法 $X \frown \sigma$ で X が σ の範疇に連接可能であることを表す.

脱落誤り修正 チャートの節点 0 と $i-1$ の間の活性弧の項 σ, 及び節点 $i-1$ と i の間の弧の項 τ に対して, 範疇 X が $X \rightsquigarrow \sigma$ かつ $X \frown \tau$ ならば, 項 $[*]_X$ をラベルとしてもつ弧をチャートの節点 $i-1$ と $i-1$ の間に追加する.

挿入誤り修正 チャートの節点 0 と $i-2$ の間の活性弧の項 σ, 及び i 番目の語 w_i の範疇 X に対して, $X \rightsquigarrow \sigma$ ならば, 項 ε をラベルとしてもつ弧をチャートの節点 $i-2$ と $i-1$ の間に追加する.

置換誤り修正 チャートの節点 0 と $i-2$ の間の活性弧の項 σ, 及び節点 $i-1$ と i の間の弧の項 τ に対して, 範疇 X が $X \rightsquigarrow \sigma$ かつ $X \frown \tau$ ならば, 項 $[*]_X$ をラベルとしてもつ弧をチャートの節点 $i-2$ と $i-1$ の間に追加する.

到達可能性は, 不要な文法規則の適用を回避するものであるが, これを誤り修正に対して用いることにより, 不要な操作を回避できる. また連接可能性を導入することにより, 脱落誤り修正, 及び置換誤り修正において考慮すべき範疇をさらに絞り込むことができる. すなわち, 先読み語の範疇に連接可能な範疇のみを考慮する.

例えば, 英語文 (4.1) において, 語 "by" が入力された時点で, "by" の直前に脱落誤り修正を実行するとき, 通常は, 図 1に示した文法に出現する 9 つの範疇すべてについて挿入を試みる必要がある. しかし, 到達可能性を利用することにより, "I think" に対する構文構造 (3.2), あるいは (3.3) の最左未決定項の範疇に到達可能な範疇, すなわち, det, gi, n, p, $pron$ のみが挿入可能な範疇となる. さらに, 連接可能性を用いることにより, 上の 5 つの範疇のうち, 先読み語 "by" の範疇 p に連接可能な範疇は gi のみであり, 結局, 図 2(a) に示すように範疇 gi の語の挿入のみ実行すればよい. 実際, 脱落誤り修正として適切なものは範疇 gi を挿入する操作のみである.

また, 語 "train" が入力された時点で置換誤り修正を実行するとき, "I think" に対する構文構造 (3.2), あるいは (3.3) の最左未決定項の範疇に到達可能な範疇は det, gi, n, p, $pron$ である. このうち, 先読み語 "train" の範疇 n に連接可能な範疇は det のみであるので, 置換誤り修正としては図 2(b) に示すように, 語 "by" を範疇 det に置き換える操作のみ実行される.

一方, 語 "train" が入力された時点で, 挿入誤り修正を実行するとき, 図 2(c) に示すように "I think" に対する構文構造 (3.2) の最左未決定項の範疇に先読み語 "train" の範疇 n は到達可能であるので, 挿入誤り修正が実行され, 語 "by" が読み飛ばされる.

このように, 到達可能性, 及び連接可能性を利用することにより, 不要な誤り修正を回避することができる.

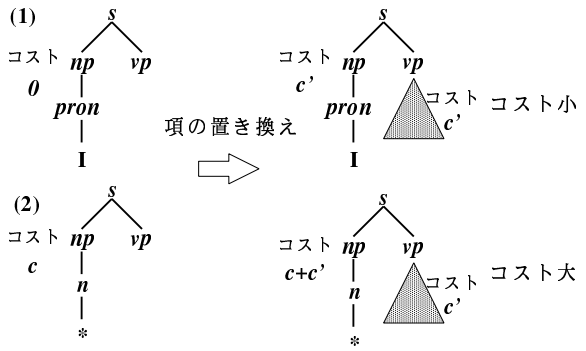

図 3: 構文構造のコストの比較

4.3 誤り修正に対するコスト

前節で述べた方法により, 文法的不適格文の解析が可能となるが, 通常の解析により作成される構文構造の他に, 誤り修正により作成される構文構造も加わるため, 全体として作成される構文構造の数は増大する. このため, 複数の構文構造の中から有用な構文構造を選択するためには, 何らかの評価基準を設けて, 構文構造の間に優先順位を与えることが考えられる. 本手法では, 入力文に対して常に誤りの存在を仮定して解析するため, 入力文中の適格な部分に対しても誤り修正を実行してしまう可能性があるが, 少ない誤り修正により作成された構文構造ほど, この可能性は低いと考えられる. そこで本手法では, 誤り修正に対してコストを導入することにより, 入力文に対する誤り修正の度合を表現する.

構文構造に対するコストの計算法は以下の通りである. まずあらかじめ, 脱落誤り修正, 挿入誤り修正, 置換誤り修正に対してそれぞれコストを定めておく. 構文構造のコストを, その構文構造の作成において実行された誤り修正のコストの和と定め, 誤り修正を経て構文構造を作成した時点でその誤り修正のコストを逐次加算することにより計算する.

例えば, 英語文 (4.1) において, 語 "I" が入力された時点における通常の解析により, "I" に対する構文構造

(4.6) $[[[I]_{pron}]_{np}[?]_{vp}]_s$

が作成され, 語 "train" が入力された時点での誤り修正により, "I" に対する構文構造

(4.7) $[[[*]_n]_{np}[?]_{vp}]_s$

が作成される. しかし, 誤り修正により作成された構文構造 (4.7) のコストに比べて, "I" を適格な部分とする構文構造 (4.6)

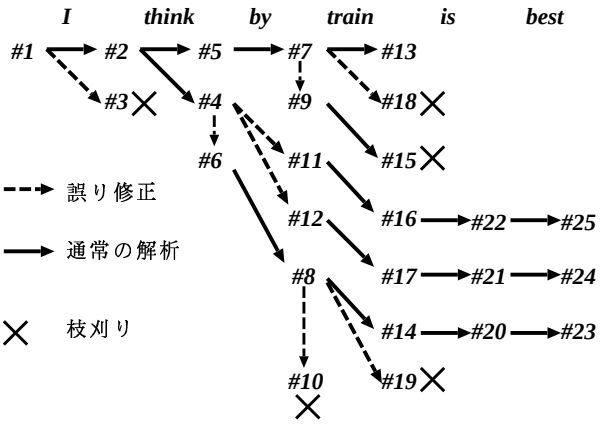

図 4: 弧の作成過程

の方がコストは低い. このように, 入力文中の適格部分に対する構造をより多く保存している構文構造を選択することができる.

また, 構文構造数の増大, すなわち弧の数の増大は解析効率の低下を引き起こすため, 弧の数を削減する必要がある. この問題を解決するために, 上で述べたコストを比較し, コストの高い弧を削除する. 弧の削除方法を以下に示す.

- チャートの節点 0 と i を結ぶ弧のうち, その項の未決定部分の範疇が一致する弧をすべて選び出す.
- 選び出した弧のうち, コスト最小の弧のみを残し, 他の弧を削除する.

例えば, 英語文 (4.1) において, 語 "I" が入力された時点で, 図 3の構文構造 (1) をもつ弧がチャートの節点 0 と 1 の間に作成される. 語 "think" が入力された時点で, 置換誤り修正により, 節点 0 と 1 の間に項 $[*]_n$ をもつ弧が作成され, その結果, 図 3の構文構造 (2) をもつ弧がチャートの節点 0 と 1 の間に作成される. 構文構造 (1) 及び (2) の未決定部分の範疇は vp と一致する. このとき, 構文構造 (1) と比べて構文構造 (2) は, 置換誤り修正を実行した分だけコストが高いため削除される. 仮に, 構文構造 (1) と構文構造 (2) に対して項の置き換えを実行する場合, 構文構造 (2) をもとに作成した構文構造のコストは, 常に構文構造 (1) をもとに作成した構文構造のコストを上回る.

この方法により, コストが最小となる見込みのない構文構造を削除することが可能となり, 効率的に解析することができる.

5 文法的不適格文の解析例

英語文 (4.1) に対する漸進的な解析処理の過程を表 1に示す.

各行がチャートの弧に対応しており, # の欄は弧の作成順, loc の欄は弧の始点と終点の対を表す. また, 弧の生成の過程を図 4に示す. なお簡単のため, 一回の誤り修正に対するコストはいずれも 1 としている.

語 "by" が入力された時点では, 脱落誤り修正により, "think" の直後に動名詞を挿入し, 弧 #6 が作成される. 語 "train" が入力された時点では, 挿入誤り修正により, 語 "by" を読み飛ばし, 弧 #11 が作成され, 置換誤り修正により, 語 "by" を冠詞に置き換え, 弧 #12 が作成される. また, 誤り修正により弧 #3, #10, #15, #18, #19 が作成されるが, これらはすべて削除される. このように, 文法的不適格文である英語文 (4.1) に対して, 入力途中の段階でそれまでの入力に対する構文構造を作成できる. なお, 最終的に作成された弧は #23, #24, #25 であり, それぞれ, 脱落誤り修正, 挿入誤り修正, 置換誤り修正を経て作成されている.

6 関連研究との比較

2 節で述べたように, 本稿で提案した手法と同じく構文的な不適格性を扱う手法は, これまでにいくつか提案されている. 本節では, 解析の漸進性という観点に着目して, 本手法とこれらの手法との比較を行う.

Mellish[15] や加藤 [8] はチャート法に基づく手法を提案している. これらは上昇型チャート解析により一文全体を解析し, 解析に失敗すると下降型解析を用いて誤り修正を実行する. しかし, 通常のチャート解析をベースとするため, 入力途中の段階でそれまでの入力に対する構文構造を作成できない. また, 誤り修正は, 文全体が入力された後で解析の失敗が明らかになってから起動される.

今井ら [4] や斎藤 [18] は GLR 法に基づく枠組を提案している. 今井らの手法は GLR 法により入力文を解析し, 解析に失敗すると後戻りして誤り修正を行う. 誤りを即座に修正できないため, 漸進的な解析に適さない. 一方, 斎藤の手法は, 入力文の解析と同時進行的に誤り修正を行う. 斎藤の手法は, 本稿で提案した手法と同様に, 解析と同時進行的に誤り修正を行う. しかし, 通常のチャート法と同様, GLR 法も入力途中の段階でそれまでの入力に対する構文構造を作成できないため, 漸進的な解析に適さない.

Lee ら [11] はアーリー法に基づく手法を提案している. 入力の解析と同時進行的に誤り修正を行い, 解析結果として項の集合を更新する. しかし, アーリー法は解析終了時になってはじめて, 全体の構文構造を作成するため, 漸進的な解析に適さない.

一方, 本稿で提案した手法は, 解析処理と同時進行的に誤り修正を実行するため, 文法的不適格文を漸進的に解析することができる. また, 漸進的チャート解析を解析処理のベースとすることにより, 入力途中の段階でそれまでの入力に対する構文構造の作成が可能である.

7 おわりに

本稿では, 解析処理と同時進行的に誤り修正を実行する漸進的な文法的不適格文解析手法を提案した. 漸進的チャー

表 1: 文法的不適格文 “I think by train is best.” に対する漸進的な解析過程

入力語	チャート				構文構造の削除
word	#	*loc*	*term*	cost	
	1	0-0	$[?]_s$	0	
I	2	0-1	$[[[I]_{pron}]_{np}[?]_{vp}]_s$	0	
think	3	0-1	$[[[*]_n]_{np}[?]_{vp}]_s$	1	削除
	4	0-2	$[[[I]_{pron}]_{np}[[think]_{vt}[?]_s]_{vp}]_s$	0	
	5	0-2	$[[[I]_{pron}]_{np}[[think]_{vi}[?]_{pp}]_{vp}]_s$	0	
by	6	0-2	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{gi}[[?]_{pp}]_{np}[?]_{vp}]_s]_{vp}]_s$	1	
	7	0-3	$[[[I]_{pron}]_{np}[[think]_{vi}[[by]_p[?]_{np}]_{pp}]_{vp}]_s$	0	
	8	0-3	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{gi}[[by]_p[?]_{np}]_{pp}]_{np}[?]_{vp}]_s]_{vp}]_s$	1	
train	9	0-3	$[[[I]_{pron}]_{np}[[think]_{vi}[[by]_p[[*]_{det}[?]_n]_{np}]_{pp}]_{vp}]_s$	1	
	10	0-3	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{gi}[[by]_p[[*]_{det}[?]_n]_{np}]_{pp}]_{np}[?]_{vp}]_s]_{vp}]_s$	2	削除
	11	0-3	$[[[I]_{pron}]_{np}[[think]_{vt}[?]_s]_{vp}]_s$	1	
	12	0-3	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{det}[?]_n]_{np}[?]_{vp}]_s]_{vp}]_s$	1	
	13	0-4	$[[[I]_{pron}]_{np}[[think]_{vi}[[by]_p[[train]_n]_{np}]_{pp}]_{vp}]_s$	0	
	14	0-4	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{gi}[[by]_p[[train]_n]_{np}]_{pp}]_{np}[?]_{vp}]_s]_{vp}]_s$	1	
	15	0-4	$[[[I]_{pron}]_{np}[[think]_{vi}[[by]_p[[*]_{det}[train]_n]_{np}]_{pp}]_{vp}]_s$	1	削除
	16	0-4	$[[[I]_{pron}]_{np}[[think]_{vt}[[[train]_n]_{np}[?]_{vp}]_s]_{vp}]_s$	1	
	17	0-4	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{det}[train]_n]_{np}[?]_{vp}]_s]]_{vp}]_s$	1	
is	18	0-4	$[[[I]_{pron}]_{np}[[think]_{vi}[[by]_p[[[*]_{pron}]_{np}]_{pp}]_{vp}]_s$	1	削除
	19	0-4	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{gi}[[by]_p[[*]_{pron}]_{np}]_{pp}]_{np}[?]_{vp}]_s]]_{vp}]_s$	2	削除
	20	0-5	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{gi}[[by]_p[[train]_n]_{np}]_{pp}]_{np}[[is]_{be}[?]_{adj}]_{vp}]_s]]_{vp}]_s$	1	
	21	0-5	$[[[I]_{pron}]_{np}[[think]_{vt}[[[train]_n]_{np}[[is]_{be}[?]_{adj}]_{vp}]_s]]_{vp}]_s$	1	
	22	0-5	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{det}[train]_n]_{np}[[is]_{be}[?]_{adj}]_{vp}]_s]_{vp}]_s$	1	
best	23	0-6	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{gi}[[by]_p[[train]_n]_{np}]_{pp}]_{np}[[is]_{be}[best]_{adj}]_{vp}]_s]_{vp}]_s$	1	
	24	0-6	$[[[I]_{pron}]_{np}[[think]_{vt}[[[train]_n]_{np}[[is]_{be}[best]_{adj}]_{vp}]_s]_{vp}]_s$	1	
	25	0-6	$[[[I]_{pron}]_{np}[[think]_{vt}[[[*]_{det}[train]_n]_{np}[[is]_{be}[best]_{adj}]_{vp}]_s]_{vp}]_s$	1	

ト解析の解析途中の段階で随時, 誤り修正を実行することにより, 文法的不適格文に対して, それまでの入力に対する構文構造を作成できることを例をもって示した.

本手法では, 通常の解析により作成された構文構造だけでなく, 誤り修正により作成された構文構造も保持するため, 結果として構文構造が爆発的に増大する恐れがある. そこで, 本稿では, この問題に対してコストを用いて構文構造数を削減する方法を提案した. より効率的な解析のために, 構文構造の作成を制御する枠組を導入することは, 今後の課題である.

著者らはこれまでに, 漸進的な翻訳処理において, 文法的不適格文に対して解析が失敗した段階で誤り修正を行うことにより, 翻訳正解率が向上することを確認している [9]. 現在, 本稿で提案した手法を漸進的な英日話し言葉翻訳システム [14] に適用することを検討している.

参考文献

[1] 秋葉 友良, 田中 穂積 : 拡張部分木を用いた漸進的構文解析, 情報処理学会第 45 回全国大会 (3), pp.175-176 (1992).

[2] 伝 康晴 : 統一モデルに基づく話し言葉の解析, 自然言語処理, Vol.4, No.1, pp.22-40 (1997).

[3] Douglas, S. and Dale, R. : Towards Robust PATR, *Proc. of 13th International Conf. on Computational Linguistics*, pp.468-474 (1992).

[4] 今井 宏樹, Theeramunkong, T., 奥村 学, 田中 穂積 : 一般化 LR 構文解析法による文中の複数箇所の誤りの検出と修正, 言語処理学会第 2 回年次大会, pp.153-156 (1996).

[5] 今一 修, 松本裕治 : 文法的不適格文処理のための統合的枠組み, 人工知能学会誌, Vol.12, No.3, pp.404-411 (1997).

[6] Inagaki, Y. and Matsubara, S.: Models for Incremental Interpretation of Natural Language, *Proc.*

of 2nd Symposium on Natural Language Processing, pp.51-60 (1995).

[7] Jensen, K., Heidorn, G. E., Miller, L. A. and Ravin, Y. : Parse Fitting and Prose Fixing: Getting a Hold on Ill-Formedness, *Computational Linguistics*, Vol.9, No.3-4, pp.147-160 (1983).

[8] 加藤 恒昭 : 一般化弧を用いた A* 探索による非文の解析, 情報処理学会論文誌, Vol.36, No.10, pp.2343-2352 (1995).

[9] 加藤 芳秀, 松原 茂樹, 浅井 悟, 外山 勝彦, 稲垣 康善 : 話し言葉における文法的不適格文に対する漸進的翻訳手法, 情報処理学会第 55 回全国大会 (2), pp.43-44 (1997).

[10] Kay, M : Algorithm Schemata and Data Structures in Syntactic Processing, *Technical Report CSL-80-12*, Xerox PARC (1980).

[11] Lee, K.J., Kweon, J.K., Seo, J. and Kim, G.C.: A Robust Parser Based on Syntactic Information, *Proc. of 7th Conf. of European Chapter of the Association for Computational Liguistics*, pp.223-228 (1995).

[12] McDonald, D.D.: An Efficient Chart-based Algorithm for Partial-Parsing of Unrestricted Texts, *Proc. of 3rd Conf. on Applied Natural Language Processing*, pp.193-200 (1992).

[13] Matsubara, S., Asai, S., Toyama, K. and Inagaki, Y.: Chart-based Parsing and Transfer in Incremental Spoken Language Translation, *Proc. of 4th Natural Language Processing Pacific Rim Symposium*, pp.521-524 (1997).

[14] 松原 茂樹, 浅井 悟, 外山 勝彦, 稲垣 康善 : 不適格表現を活用した漸進的な英日話し言葉翻訳手法, 電気学会論文誌, Vol.118-C, No.1, pp.71-78 (1998).

[15] Mellish, C.S.: Some Chart-Based Techniques for Parsing Ill-Formed Input, *Proc. of 27th Conf. of Association for Computational Linguistics*, pp.102-109 (1989).

[16] 中野 幹生, 島津 明 : 言い直しを含む発話の解析, 情報処理学会研究報告, SLP-16, pp.1-6 (1997).

[17] 佐川 雄二, 大西 昇, 杉江 昇 : 自己修復を含む日本語不適格文の分析とその計算機による理解手法に関する考察, 情報処理学会論文誌, Vol.35, No.1, pp.46-52 (1994).

[18] 斎藤 博昭 : 一般 LR 構文解析法におけるエラー処理, 情報処理学会論文誌, Vol.37, No.8, pp.1506-1513 (1996).

[19] 山本 幹雄, 小林 聡, 中川 聖一 : 音声対話文における助詞落ち・倒置の分析と解析手法, 情報処理学会論文誌, Vol.33, No.11, pp.1322-1330 (1992).