7章　上向構文解析

スライド1

　第5章と本章で扱う「構文解析」は，コンパイラのなかでは最も理解し難い部分のようです．しかし，その基本をなす考え方は極めて常識的で合理的です．したがって，本章を読み進めるにあたっては，細部にとらわれることなく，大まかな考え方がそれとなく理解できることに努めてください．これができればしめたもの，すでに80%は理解できたも同じです．それでは先に進みましょう．

私たちは第5章において，生成規則と先頭記号集合を用いた下向構文解析を勉強してきました，下向構文解析は開始記号から始め，次に入力するトークンを先頭記号集合にもつ規則を探索しながら，適用できる可能性のある規則を順次特定していくものでありました．そこでは，構文木は上（開始記号に相当する根）から下（トークンに相当する葉）に向かって作られていきました（第5章スライド14を参照）．

　これに対して，この章では，現在までに生成規則のどこまでを処理し次にどこを処理するかを状態にもつオートマトンを予め用意しておき，そのもとでトークンを読み取るたびにこのオートマトンを使って構文木を下から上に向かって作っていく上向き構文解析について学ぶこととします．

スライド2

　最初に上向構文解析を定義してみましょう．上向き構文解析とは，トークン列である記号列が生成規則の右辺と一致したとき，左辺の記号に置き換えることを繰り返しながら構文木を作成していく解析法を言います．この定義からも明らかなように，この上向構文解析においては，右辺から左辺に，すなわち，構文木は下から上に向かって作られていくことになります．

　上向構文解析の特徴としては，スライドにもあるように

・上向構文解析はその適用の広さから現在のプログラミング言語の構文解析法の基本となっている．

・上向構文解析は解析の進行具合を表すオートマトンと，未探索な要素を記憶したスタックを用いて実施される．

・上向構文解析が扱うことができる文法のクラスは下向構文解析のそれより広い．

などが挙げられます．

スライド3

　上向構文解析の代表的な解析法にLR構文解析があります．これは，それぞれの生成規則が現時点までにどこまで解析されたかを表す状態をもつオートマトンを予め用意しておき，トークンを読み取るごとにそのオートマトン上で状態を遷移させながら構文木を構成していくものです．そこでは，構文木の要素が探索されるたびにそれをスタックに記憶し， 探索がある段階まで進むとスタックに記憶された要素にバックトラックし新たな探索が行われる，という手順が開始記号に至るまで繰り返されることとなります．探索は，木の要素をスタックに積むように左から右に並べ（Left to right），LIFO（後入れ先出し）の規則に従って右から木が構成されていきます（Rightmost derivation）．LR構文解析のなかでも，最小限の基本的な動作のみを行うLRはSLR（Simple LR）と呼ばれています．

スライド4

　SLR構文解析の考え方を理解するために，スライドにあるような生成規則をもつ文法を考えてみましょう．この文法にはいかなるあいまいさもないことに注意しておきましょう．もし，文法にあいまいさがあると，第4章で示したように構文木が一意に定まらず，自動的に構文解析を行うことができなくなります．

　次に，それぞれの非終端記号Hについて，このHの直後にくる可能性のある終端記号の集合をHの後続記号集合と呼び，以下ではFollow(H)で表すこととします．なお，この例のなかの $は入力されたトークン列の最後を表す終端記号を表すものとします．後続記号集合の一例として，今，このスライドに示した生成規則の例のなかの非終端記号Tに着目すると，

より，Tの後続記号集合は

であることが確かめられます．

スライド5

　次に，生成規則において，現在までに処理した時点を示す黒い点・，これをLRマーカーと言うこともありますが，この・を付加した生成規則をLR項目と呼ぶこととします．例えば，このスライド左下には，規則に対して，4つのLR項目が記載されています，最初のLR項目はまだどの記号も処理されていないことを，2つ目のLR項目はEが処理されて+が処理される前であることを，3つ目のそれは+までが処理済みであることを，最後のそれはTまでのすべてが処理されていることを示しています．

また，LR項目集合*A*に対して，*A*に含まれる任意のLR項目において，・のすぐ右にある記号が非終端記号の場合，その非終端記号からの生成規則で・を右辺の最左部においたすべてのLR項目を含む集合をAの閉包と言い，で表すこととします．例えば，の閉包はスライドの右下のようになります．分かり易く言えば，“Aの閉包とは，Aに含まれるそれぞれのLR項目に対して，次の時点で選択される可能性のある規則とそこで処理される可能性のある記号のすべてを表したもの”と言うことができます．

スライド6

さらに，開始記号から始めて，スライドに示された手順(1)～(3)で作られる項目集合を正規項目集合と言います.

(1) 開始記号をもつ生成規則の右辺の左端に・をおいた項目だけから作られる項目集合の閉包をとします． はまったくトークンを処理していない状態を表します．

(2) に含まれる項目について，・の直後の記号（終端記号，非終端記号ともに）1つが同じものをまとめ，・を右に1つ移動します．それぞれについて閉包をとり，項目集合を作ります．

(3) 以下(2)と同様に，すでに作られた項目集合から， ・の直後の記号1つが同じものをまとめ，・を右に1つ移動しながら新たな項目集合を作ります．

これらの例が，教科書pp.76-77のからに相当します．それぞれについて，そのようになることを確かめておいてください．

　正規項目集合も先の閉包と同様に分かり難い概念であるかもしれません．しかし，その内容はいたって簡単なものです．これも分かり易く言えば，“正規項目集合とは，各時点で選択可能なすべて規則の集合”を表しており，“からを作成するとは，ある時点での正規項目集合のなかから1つのLR項目と次に処理する記号を選択することによって，次の時点での正規項目集合が決まること”を表しています．

『余談』

この種の記述を正確に読み取るにあたっては，一字一句を確かめながら（助詞，助動詞，形容詞，副詞，連体詞の使い方も含めて）文章を正確に読み取る癖をつけておいてください．中学・高校で使った教科書は，正しい表現に徹した最も身近な手本の1つです．

スライド7

　これで，LR構文解析に使われるLRオートマトンを作成する準備ができました． LRオートマトンの作成方法をこのスライドにまとめます．

すなわち，

＊状態

　それぞれの正規項目集合に対応した状態を作成する．

＊入力とアーク

項目集合から，・の直後の記号をまとめ・を右に1つ移動して新たな項目集合が作られたならば，状態から状態至る入力をもつアークを作成する．

ことによって，LRオートマトンである状態遷移図が完成することとなります．

スライド8

　スライド4に示した生成規則で表される文法に対して，今の手順で作成されたLRオートマトンである遷移図をこの図に示します．状態はからの14個の正規項目集合から構成されています．一方，各状態からのアークは，それぞれの状態に含まれるLR項目において・の直後にある記号の種類の個数だけのアークから構成されています．例えば，状態には8個のLR項目が含まれており，各LR項目の・の直後の記号にはの6種類の記号があります．したがって，遷移図にもあるように，からはこの6種類の記号をそれぞれラベルにもつ6本のアークが描かれています．スライドの遷移図はこのようにして作成されたものです．

スライド9

　SLR構文解析は，今示した遷移図を使って，トークンを読み取るごとに進められていきます．そこで，次にその動作について示します．

スライド10

　SLRの一般的な動作を形式的に説明するより，先ほどの例を使って，まずはその動作を具体的に説明する方が分かり易いと考えられます．そこで，先ほどの例を使って，スライドのなかの図7.1と図7.2を見ながらその探索過程を追ってみましょう．入力列は

　　　入力列（トークンの並び）：2+3\*5

です．

①状態は初期状態I0．入力列の最初の記号は2（2はn(number)）より状態はI4に遷移．

②I4からはこれ以上状態が遷移できないため，I4からI0にバックトラックし，あらためてI4のLR項目の左辺であるFを入力として考えると，状態はI0らI3に遷移．

③I3からもこれ以上状態が遷移できないため，I3からI0にバックトラックし， I3のLR項目の左辺であるTを入力として考えると，状態はI0らI2に遷移．

④I2からも次の入力が+であることよりこれ以上状態が遷移できないため，I2からI0にバックトラックし， I2のLR項目の左辺であるEを入力として考えると，状態はI0らI1に遷移．

⑤I1からは次の入力+に従って，状態はI7に遷移．

⑥I7からは次の入力3(n)に従って，状態はI4に遷移．

⑦I4は行き止まりのためI4からI7にバックトラックし， I4のLR項目の左辺であるFを入力として状態はI7らI3に遷移．

　　　・・・

以下，同様の手順を繰り返すと，スライドのなかの図7.2が得られます．

最後まで，図7.2を見ながらその手順を追ってみてください．この手順のなかで，新たな状態の保持とバックトラックに以前に勉強したスタックが用いられています．

スライド11

　このスライドは以上の動作を一般的な形でまとめたものであり，教科書に記載されているものです．

スライド12

　以上の構成法は，結局のところ，構文木をトークン列から予想しながら下から上に向かって作り上げていくことの相当します．パワーポイントをクリックを続けながら，このことを確かめてください．このように，第5章で勉強した下向構文解析は構文木を上から下に向かって構成するものでしたが，この章で勉強した上向構文解析はそれを下から上に向かって構成するものであることが確かめられたと思います．

スライド13

　この節7.2の最後に，以上のような動作を参考にまで表形式の形でまとめておきます．

スライド14

　これまでに私たちはSLRの動作を見てきました．そこでは，LRオートマトンを使って，次にくる1つの記号を見ながら解析を進めてきました．しかし，生成規則によっては，これまでのSLRでは，その動作が一意に決まらないこともあります．以下では，このことについて少しだけ触れておきましょう．

スライド15

　あらためて，これまでの構文解析は，状態0から始めて，現在の状態と次にくる終端記号あるいは直前に簡約化した非終端記号をもとに，どの状態に遷移するかを調べるものでした．したがって，この解析が実行できるためには，ある状態でその動作が唯一に決定できることが，すなわち簡約か積み上げか，言い換えれば次のトークンを選択するか，あるいはバックトラックするかを一意に決定できることが必要となります．しかし，規則の形によっては，これができないこともります．このとき，この文法は1個の記号を先読みするSLR構文解析では解析できないため，SLR文法ではありません．

　そこで，SLR文法でない文法に対応できるよう，次にくるk個の記号までを先読みすることを考えてみましょう．このような構文解析をSLR(k)構文解析と言います（これらはちょうど第5章で学んだLL(k)構文解析に対応するものです）．これに対して，これまで勉強してきた1個だけの記号を先読みしてきたSLR構文解析を，SLR(1)で表すこととします．このとき，このスライドにもあるように，言語の記述能力は

であり，より多くの記号を先読みするほど多くの言語を処理することができることが知られています．このことは，以前に勉強したようにLL構文解析では，でしたが，これとは大きく違うこと，すなわちSLR構文解析では先読み効果が期待できることが分かります．なお，参考にまで，LL構文解析とSLR構文解析の間には，

, は先読みを行わないSLR構文解析

の関係があることも知られています．これらはいずれも，理論的にも実用的にも極めて興味深い事実です．しかし，授業ではこれ以上深入りしないでおきましょう．

　以上で，私たちは上向構文解析の代表として，SLR構文解析の考え方とその動作について勉強してきました，また，その拡張としてのSLR(k)についても勉強してきました．上向構文解析についてはSLRのほかにもいくつかのものがあります．しかし，その基本的な考え方はいずれもSLRと同じです．このため，7.4節および7.5節は省略し，第7章の学習はここまでとします．

　最初にも示したように，ここでは上向構文解析の動作が大まかに理解できれば，それで十分です．機会があれば，構文解析の要点を簡単に説明できればと考えています．