修 士 論 文

CafeOBJ のラーニングシステムのデザインに関する研究

指導教官　二木厚吉

北陸先端科学技術大学院大学
情報科学研究科

市山洋乃

平成11 年 8 月 13 日

要 旨

形式仕様とは、健全な数学的基盤を持つ記述言語で書かれた仕様のことである。そのため、あいまいな表現になりにくく、機械的な検証ができることがおおい。

大規模、複雑でなおかつ信頼性が要求されるシステムに対し、形式的に仕様を記述し、部分または全体の検証を行い、システム全体の信頼性の向上を図ることが試みられている。

そして、共同で作業を進めていくことを考えた場合、厳密な仕様の記述ができるということは、それを提示された側が、それに対する誤解をしにくいということが言えるだろう。そういった記述方法を学ぶために、今までいくつもの教科書が作られている。

一方、計算機に人間が学習することの支援をさせようという考えは、さして新しいものではない。だが、最近までは、計算機性能の限界や、コストのために、あまり現実的な試みとは言えなかった。

しかし、今日の計算機の発展により、かなりのことが可能となってきている。

印刷し、本のかたちにまとめた教科書にも数々の利点がある。だが、計算機で学習の支援を行うことについて、比較検討を行った結果、十分な利点があると言える。

対象を、あまり形式手法になじみのない人間とし、題材を実行可能な仕様記述言語であるところのCafeOBJにし、ラーニングシステムを製作した。

内容は、仕様記述に関する解説から始め、例題を基本的なデータ構造に取り、例題同士の類似性を強調することで、理解を促し、実際に学習者が自分で記述できるように解説し、演習のタイミングを示している。そして、学習を継続させるために、学ぶ側の人間が迷わないよう、全体の色使いや、一度に表現するテキストの分量などを工夫した。

目 次

第 1 章

研究の背景

本研究は、形式仕様記述言語 CafeOBJ の学習を支援するラーニングシステムのデザインに関する研究である。

健全な数学的基盤に基づいた手法を、形式手法といい、その基盤は形式仕様言語で与えられる。CafeOBJ は順序ソート代数を意味論の基盤にもった、仕様言語である。

形式手法で書かれた仕様は、健全な数学的基盤を持つために、曖昧な記述になりにくい。そのため、多人数で仕様をやりとりすることを考えた際に、仕様に対する理解の食い違いがおこりにくくなる。

ある程度以上、大きなシステムを作ることを考えた場合、仕様を多人数でやりとりすることは不可欠である。

すでに、形式手法に関しては、いくつものドキュメントが存在する。CafeOBJ の特徴のひとつである、実行可能であるということをいかすこと、及び、解説の構造をハイパーテキスト構造にしたいがために、計算機で学習の支援を行った。

あることがらに対して、いくつかの解説が存在する場合を考えてみる。または、ある事柄について、読者の状態によって、参照するべきであったり、しなくても良いテキストがあった場合についても同様に考えてみる。テキストがうまく電子化されていれば、ハイパーテキスト構造にして、問題を処理することは容易だ。しかし、紙に印刷され、本のかたちになっていた場合は、読者側にとって、そういった参照のしかたは、ストレスになる。

さらに、図などを読者側が参照したいと思ったときのことを考える。読んでいる文章の近くにあるものであったとしても、物理的に紙の大きさがあるため、紙媒体が利用されていた場合は、必ずしも同時参照が出来るとは限らない。

紙媒体に印刷される予定のものであっても、電子化されたテキストは、後から修正が可能であるという特徴をもっている。しばしば修正されるテキストは、学習者の側に不信感をうえつけることにもなりうる。しかし、修正がまったく出来ないのと比べた場合、十分な期間をおいて修正が行われるのであれば、やはり修正可能であるということは、有用である。

さらに、CafeOBJ は実行可能な仕様記述言語である。テキストが紙であった場合であっても、CafeOBJ

が利用可能な状態で学習を行うであろうことは十分予測出来る。そのため、計算機で支援を行うと、あたりまえのことながら計算機のある環境でないと利用できないというデメリットは、あまり目立ったものではないだろう。

電子化して、計算機上でテキストを利用することを考えた場合、目が疲れる、環境が必要であるなどの問題点は存在する。しかし、メリットとデメリットを比べた場合、十分に、計算機で学習の支援を行う意味があると思われ、システムを製作することにした。

第 2 章

形式仕様

この章では、形式仕様と、本研究の題材となった代数型仕様言語 CafeOBJ について説明する

2.1 形式手法

形式手法は、システムのあいまいさや不完全さ、矛盾をあきらかにするために使われる手法である。

多人数であるシステムを作る作業をする際においては、そのシステムに関する仕様をやりとりすることが必要になってくる。非形式的、形式的にかかわらず、色々な仕様記述の方法が存在し、それぞれのメリットを持っている。

もし、ある手法が健全な数学的基盤を持つならば、その手法は形式手法と呼ばれる。形式手法であるところのメリットは、非形式的な手法に比べ、一般的にあいまいさが少なく、正確で簡潔な表現ができることである。数学的基盤に基づいて、一貫性や完全性が正確に定義されるため、機械的に検証することができる。

形式手法は、全体、時には部分部分の正しさを証明することで、システム全体の信頼性の向上に貢献することもできる。

手法の数学的基盤を与えるのが、形式仕様言語である。そして、仕様言語としては、集合論をもとにした Z、VDM、代数をもとにした Larch、OBJ、CafeOBJ などが有名である。

自然言語や図などを用いた非形式的手法は、一般的に形式手法にくらべ曖昧であることが多い。だが、その曖昧さゆえに、直感的に理解しやすい、または、実際に作る人間の裁量に任せる部分を表現しやすいとも言われている。

しかし、あいまいな表現ができるというのと、あいまいな表現であるというのは、意味が違っている。大規模で複雑なシステムを構築する場合において、多人数で仕様をやり取りするのは不可欠だろう。そして、多人数で仕様をやり取りすることを考えた場合、なるべく全員の理解を一致させるというのは重要なことである。形式手法の、あいまいな表現になりにくいというのは、非常に大きなメリットになる。

2.2 CafeOBJ

CafeOBJ は、代数仕様言語の OBJ の特徴を継承した、実行可能な仕様言語である。そして、基盤として、順序ソート書き換え論理を持っている。

OBJ からは、豊富なモジュールの構成方法、パラ一メータ機構、順序ソート代数を意味論の基盤として持つ、実行可能である、項の表記が自由に出来る、などの特徴をうけついでいる。

それに加え、内部構造を持ったデータ型を勘弁に記述するための構文、動的な振る舞いを記述するための構文、オブジェクトプログラミングの技法などを新たに取り入れている。本研究では、ラーニングシステムの対象となる仕様言語を CafeOBJ としている。その理由は、以下のような CafeOBJ の特徴が、学習を行うにあたって、学習を継続する動機をもたせたり、学習者に現状に対する正しい認識を持たせるのに、有用だからである。

- 書いた仕様を実行可能である
- モジュールの輸入が行える
- 項の表記が自由に行える

書いた仕様を学習者に実行させることで、達成感を持たせることができる。さらに、学習者が書いた仕様について、少なくとも文法的に正しいかといったチェックができるので、自らの理解に対する認識を形成する材料とさせることができる。

モジュールの輸入ができるため、先に書いた仕様を利用して、別のモジュールを書くといったことがしやすい。このことで、学習の結果を、実行可能であることとともに、学習者に見せることができ、達成感につながる。学習を継続する動機を続けて持たせるためには、ある程度の達成感を与えることが必要不可欠である。

CafeOBJ では、前置、中置、後置のような標準記法をはじめ、演算名を構成するトークン列の任意の位置に引数を置くことができる。表記が違っていても、同じ演算ならそうであると学習者に理解させたいのは確かだ。しかし、学習者を、形式手法に馴染みのない人間としてラーニングシステムを製作するため、学習者が仕様記述言語そのものに対して以外の場所で戸惑うことを極力さけたい。そのため、学習者になじみがあると想定した自然言語などで記述された図と、おなじかたちで項の表記ができるということは、学習者の新しい知識にふれたときの戸惑いを減らすために有効であると思われる。

以上の特徴から、ラーニングシステムの対象とする仕様記述言語を CafeOBJ とした。

第 3 章

ラーニングシステム

この章では、様々なラーニングシステムについて説明する

3.1 解説指導型コースウェア

学習内容の説明の後、学習課題が提示され、それに応じた回答群がしめされ、学習者がそのひとつを選ぶと、それに応じて正誤やヒント、次の問題などが続き、個人の理解に応じて学習を進めることができるもので、ほぼ現在のラーニングシステムの原型となるシステムである。

解説指導型コースウェアと呼ばれるものでは、完全に教授側が学習者のたどりうる道筋を決めてしまわなくてはならない。ハイパーリンク型のように、必要と思った学習者が能動的にコースを選ぶわけではなく、学習者のテストに対する反応から、コースが選択されるだけである。

これでは、システムの複雑さが、用意したコースの数で大きく違ってきてしまうし、学習者側の個別の要求にも応えられるとはいいがたい。

だが、解説を提示し、理解を確かめるための試験をし、それに対する反応で学習者に提示する内容を決めるというのは、多くの教育システムに共通する事柄であり、やはりこのシステムは、現在のラーニングシステムの原型と言える。

3.2 知的教育システム

知的教育システムとは、教師としてのコンピュータと学習者との間の適応的で双方向な知識交流の実現を目指して作られたシステムであり、教材モジュール、学習者モデルモジュール、教授戦略モジュール、インターフェースの 4 つのモジュールからなる。

教材モジュールには、教授する内容だけでなく、学習者の解答の評価、問題解決、問題生成、説明のための情報の提供などの機能を持っている。学習者モデルモジュールは、学習者のふるまいから、学習者の教材に対する理解のモデルを想定し、他のモジュールにその情報を提供する機能を持っている。そして、教授戦

略モジュールが、それらの情報から問題の選択などの方針を決定する。

いわゆる CAI システムでは、教材の各項目に対する提示情報とその制御をひとつの単位として構成し、教授項目のながれにそってそれを提示していくという方針が採られている。そのため、システムの自由度が、分岐の数そのままとなり、かなり堅い制御しか行うことが出来ない。

その欠点を克服することを目指して、ITS(Intelligent Tutoring System)、ICAI(Intelligent CAI) などと呼ばれるシステムが作られている。

CAI システムとこれらのものの大きな違いは、教材とその他の手続き的知識にシステムがはっきりと分割できるかどうかであるといえる。

そして、現在では、さらに学習を効率化するために、教師と一人の学習者というシステムではなく、複数の学習者の間の交流というかたちで、学習をすすめていくものや、学習者が教師の立場に立ち計算機が用意した生徒に教授することで学習の定着を図るものなどが作られている。

3.3 ハイパーリンクを利用したシステム

教育の場において、計算機を利用するとき、従来利用してきた教材の電子化が考えられる。しかし、この際、テキストをそのまま電子化しただけでは、紙の持つ場所を選ばないなどの利点を失うことになり、デメリットのほうが大きくなってしまう。

学習教材に おける教授知識は、文章の流れによって表現されるだけでなく、文章や図面、文章間などに相互参照として表現されており、教授するがわの指示により参照されたり、説明されたり、文章中に参照すべき項目が示されていたりする。これらの複雑な相互参照の関係は、文章の流れを乱すため、紙メディアでは表現しづらく、また複雑な相互関係は本来の流れを見失いやすく、利用しづらい。このような関係の定義と利用を計算機によって支援することによって、電子メディアに移行するメリットを見出した。[福島 93,]

実際のシステムは、基本的にはいわゆる解説指導型コースウェアにハイパーテキストの考え方を利用したものだった。システムを構築して評価した結果として、画面表示上の問題点と、ハイパーテキスト型そのものの問題点があった。

このシステムでは、科目学習手順の指導を行うためのコースアウトラインモードと、実際に学習を行う単元モードの 2 つのモードにわかれていた。そして、ノードの前後関係、内容関係、アウトラインの関係の 3 種類のリンクを設定する。また、教科書における、章、節、項などに対応するために、コースノード間の関係からなるコースグラフをコースノードとして包含することを可能としている。

ハイパーテキスト型の問題点としては、学習教材の中のリンクが、どういった性質のものかわかりにくいという問題が発生した。内容に関するリンクと学習レベルに関するリンクがわからなくなると、学習者がより早く先に進もうとしたとき、リンクがあるのにそれに気づかないために早く進めない、逆に、内容似ついていけず、より詳しい説明を求めているのに、どこに向かえばいいのかわからなくなり、結局学習意欲を失ってしまう、などの問題が発生した。構造に関するリンクと、スケジュールに関するリンクがわからなくなると、その章で何を学ぶことが目的かわからず課題をこなすだけになってしまう、なんとなく書いてある

ことを流してしまうなどの問題点が発生した。

したがって、学習を進めていく上で必要なリンクが揃っていても、表現がわるければ、結局のところ、学習者の混乱をまねくだけであり、ハイパーリンクの利点をいかしきれないことがわかった。

ただ、ここで例に出したシステムは、WWW が普及する前のものであって、問題としている部分が、かならずしも現在問題になっているわけではない。ハイパーテキスト化しさえすれば、問題が解決するとしているように見える部分すらある。

これにならった方法論でハイパーテキストの内容を提示した場合、学習者側に見ることを要求している場所が多すぎ、結果として重要な情報が省みられない、混乱してしまうなどのことも考えられる。多くのウィンドウで、学習者が必要であろう知識を先回りして全部表示するようなやりかたは、学習者側に与えられる情報が多すぎ、提示された情報を総て理解しきれないなどの問題が起こりうる。

ただ、特に注意しなくてはならない部分として共通なのは、学習者が学習の流れのどの部分に居るかわからなくなってしまうことや、リンクの意味がわからないといったことに注意すべきであるという部分だろう。

3.4 実習支援機能を持つ教育システム

プログラミング教育において教授しなくてはいけない事柄は、そのプログラミング言語の言語の基礎となっている概念から、実際のコーディングテクニックにいたるまで、たくさん存在する。さらに、プログラミングの能力は机上の教育だけでは十分に身につけることが難しく、ほとんどの場合、課題を実際にコーディングしてみたりすることにより、机上で得た知識を実際に適用する際のやり方を学ぶための、プログラミング実習が不可欠である。

それらのことから、実習支援機能もつ、CAI のシステムが開発された。

対象言語の処理系を改造し CAI のシステムと密に融合させる、CAI のシステムのなかに対象言語の処理系のシミュレーターを組み込むなどの方式が考えられたが、実際にはマルチプロセス機能を利用して、対象としたエキスパートシステムツールおよび論理型言語のインタプリタと、CAI のシステムを並列に実行し、両方のプログラムを連動する方式が取られた。[芳賀 93,]

実際の方法としては、CAI のツールと各言語のインタプリタの間でファイルを共有し、教材のソースプログラムの部分にインタプリタが直接解釈実行できるコードを表示し、学習者が直接編集できるようにしてあった。

3.5 WWW を利用したシステム

ラーニングシステムにWWW を用いることにより、学習者がわの制限をずいぶんと減らすことが出来る。

いままでは、ある特定の計算機がある環境でないとそのシステムを利用した学習が出来ないという状況であったものが、WWW を利用することにより、ハードウェアや OS の違いを超え、ネットワーク環境があるところでならいつでも学習ができるというところまで制約を緩めることが出来た。

そのほか、簡単にテキストをハイパーリンク構成にできること、音やアニメーションなどのマルチメディアを利用できることなども、利点にあげられる。

だが、単に学習事例を提示しただけでは、WWW ベースであることの利点を生かしきっているとは言えないだけでなく、現在の環境整備からの欠点、通信容量の制限などや、個々のユーザーの環境の違いなどからくる読みにくさなどが目立つだけで、あまり有用であるとは言えない。

WWW の特色をいかし、インタラクティブ性をもたせた、隠蔽代数に関する教育システム [Joseph 98,] や、認知心理学の諸概念を適応た、大学院修士過程向きの講義であるところの「ヒューマンインタフェース論」に関するラーニングシステムが教育システムがある。[大林 98,]

WWW ベースになるに至って、ハードウェアや OS の違い、場所的な制約などがずいぶんと減った。その代償として、利用する側の環境が、特定できなくなってしまった。

従来のシステムであれば、利用する側は、特定の計算機に向かって、そのシステムを利用していた。しかし、WWW ベースで、その利点を生かした利用の仕方をしようとすれば、あまりに想定される利用する側の環境が多すぎることになる。ほぼどんな環境にも対応する機能といえば、ハイパーリンク構造くらいになってしまう。

学習者が利用する OS の違いによる画面環境の違い、ブラウザの違いによる機能制限、ハードの違いによる解像度や色数の違いなどに対する配慮は、いままでの特定の環境のみを想定していた場合には、なかったものである。さらに、学習者が同じ環境で利用していても、ブラウザのおおきさをかえているだけで、一画面に収まりうる情報量は変化し、与える印象もずいぶんと違ったものになってしまう。

完全に、学習者側の環境を限ってしまえば、紙媒体並のレイアウトも可能になってくるし、マルチメディアの利用も、完全に想定した通りのものが利用可能である。しかし、それでは、従来の環境が完全に固定されたラーニングシステムと変わらなくなってしまう。

結局、環境を選ばないというのは、かなり誇張した言い方である。対応する環境は多いが、どれだけの環境に対応することを想定するかなどの選択が、製作者がわにとって重要になってきたにすぎない。なまじ、想定した以外の環境でもある程度の学習はできるため、学習者がわに、必要とする最低限の環境の通知は必要と思われる。

第 4 章

CafeOBJ のラーニングシステムの設計

どこまで解説するか、例題の長さはどこまで大丈夫かといったことを決めるために、利用者を想定しなくてはならない。わかっていることがらがくりかえし説明されていることも、解説が少なすぎることも、利用者の学習意欲をそぐことになるためである。さらに、ハイパーリンク構造を利用するつもりでいるため、あまりにもリンクが多すぎれば、利用者が自分の居る位置を見失うような場合がありうる。そのため、利用者のレベルを想定することは、不可欠であり、本研究ではそれを、形式手法にあまり馴染みのない人間とした。1 度に学習者に提示するテキストの分量や、色彩を用いた直感的理解を促す構造、学習内容の定着を促すための例題の構造などが、特徴となる。

4.1 継続の動機を持たせるための構造

学習を継続する動機として、肯定的なメタ認知はとても重要な要素のひとつである。

メタ認知とは、自分がどれだけ学習内容を理解しているかということに対する理解のことである。否定的なメタ認知は知的好奇心を抑制し、肯定的なメタ認知は誘発する。問題が解決できなかった場合に、原因の推測如何では、否定的なメタ認知を持ってしまうことになり、学習継続の動機に悪影響を与える。

こういった、否定的なメタ認知を、肯定的なメタ認知に転換させることができれば、知的好奇心が誘発され、学習が急速に進むこともありうるそのため、いかに学習者に肯定的なメタ認知を持ちつづけさせるか、または、いかに否定的なメタ認知を肯定的なものに転換させるかが、重要になってくる。

メタ認知は、他者との関係により形成されていく部分もある。他者の自己への評価から、自分がある課題に対して、どういったことを行っているか、全く的外れなことをやっているのか、それとも正しいことをやっているのかなどから、自分の能力に関する認知を作り上げていく。教授してくる相手に対する信頼感で、フィードバックのされかたは違ってくる。知識を与えてくる対象が、学習者にとって権威であれば、あまり吟味せずに受け入れることが多い。そして、肯定的なメタ認知は、適度な達成感の連続で与えることができる。そして、学習の過程で、自らのメタ認知に疑問を持つようになった場合、既存のメタ認知が意識に上り、それが調整に影響を与える。

このシステムでは、肯定的なメタ認知をもたらす達成感をひとつの目標としている。そのために、以下のようなことを行った。

- 学習者にはなるべく多くの回数、実際に仕様を記述させること
- 例題のデータ構造はごく基本的なものをもちいること

CafeOBJ は実行可能な仕様記述言語である。そのため、実際に仕様を記述し、実行することにより、労せずして、他者からの評価を得ることができる。システムからの応答ということで、文法を評価したものとしての信頼感は、随分と強い。信頼できる他者からの評価を、こまめに受けることで、正確なメタ認知を得ることができ、肯定的評価から、肯定的なメタ認知を得ることができる。

課題そのものに、学習者が馴染みを持っていれば、メタ認知の調整にいい影響があると考えられる。そのため、CafeOBJ の記述にしろ、それを利用した検証にしろ、ごく基本的なデータ構造を用いた。これは、記述する例題を、ごく基本的なデータ構造にすることで、例題そのものにおける理解を期待し、メタ認知の調整に役立てることを狙ったものである。

そういったやり方を用いて、学習を継続する動機を持たせるよう、システムを構成した。

4.2 理解の手がかり

スキーマとは、人間の記憶のかたちとしてのデータ構造を表す言葉である。対象となるもの、場面、行動のしかたなど、それらをまとめた記憶の構造をスキーマと呼ぶ。[御領 93,] 理解しているというのは、いわば、ある事柄に対して、適切なスキーマをあてはめることができるということと言え、その段階は以下のことがらのくりかえしと見ることができる。

- 知識を収集、つまり、新しいスキーマを学習者の記憶のデータベース内に蓄える
- 既に獲得しているスキーマを調整する
- スキーマの使用に熟練する

そういった考えに基づいて、理解できないということについて考えて見る。まずは、学習者が、全く出来事に対するスキーマを持っていないという場合が考えられる。そして、学習者が適切なスキーマを持っているとしても、正しく適応できないという場合もありうる。

後者の場合は、文脈や視点を与えることがスキーマを呼び出すヒントとなり、学習を促進できる。

前者の場合であれば、まず、学習者が新しいスキーマを作る手助けをしなくてはならない。全く未知のことがらについての解説を読むことで、全く新しいスキーマを作ることもできる。しかし、それはメタ認知の正確さに関しても含めて、かなり困難である。全く新しい概念を形作らせるのではなく、学習者がすでに持っているスキーマに対して、類似したものや対比するものを提示し、そこからあたらしいスキーマを作らせることもできる。

このシステムは、形式手法にあまり馴染みのない人間を対象にしている。そのため、もともと学習者が適用すべきスキーマを持っているとは考えにくい。新しいスキーマを形成する手助けをする必要がある。

様々な角度からの説明をあたえることで、学習者がすでに持っているスキーマと、新しく獲得しようとしているスキーマを結びつけることを狙った。図を見せる、自然言語による説明をいくつか用意しておく、別の例えば手続き型のプログラミング言語で題材を書くなどのことを行うことにより、学習者がもともと持っていたスキーマを利用することを狙った。さらに、繰り返しよく似た例題を提示し、実際に行わせることで、リハーサルによるスキーマの構造化をうながし、学習結果の定着を図った。

4.3 提示する情報

人間は、ある程度以上の情報をいちどに提示されても、そのすべてを理解することはできない。短期記憶におかれたものはリハーサルにより長期記憶に移される。短期記憶に入りきらない分量のものを提示されても、長期記憶に移す前に処理しきれない。

短期記憶の容量に関する、さまざまな研究が存在する。一般的にそういった研究は、簡単な数字などについて行われていることが多い。そのため、文脈の存在するものにそのままあてはめることはできないかもしれない。しかし、人間がいちどに処理しきれる情報量に限りがあるということがらは、考慮しておくべき事柄だと思われる。

CafeOBJ の解説の場合、とくに検証の部分においては、似たような文がならび続けるものを見ることになる。そういった部分をいちどに見せずに、ある程度のまとまりで区切って学習者に見せることを行った。ただし、学習者が以前に見た部分を見たいと思った場合には、つながりのほうが重要であるとして、今まで表示したものをすべて見せてある。

具体的に言うと、検証においては補題ごとにくぎって新しいものを表示している。仕様記述に関しては、例となる仕様があまり長いものではないため、全て一度に見せてある。もっとも、次で述べる色彩がもたらす効果を利用し、signature 部と axiom 部にわけて、情報のかたまりを示してある。

形式手法に限らず、新しい内容を学習することを考えた場合、妥当な情報のかたまりを学習者が認識することは困難であり、重要なことである。そのため、システムがわからむしろ強調しすぎであるというくらいに、小さな情報のかたまりを学習者に見せた。

4.4 色彩がもたらす効果

色は、感覚的主観的なものであり、経験や文化によっても印象が違ってくる。だが、色から感じる印象や意味とは別に、見やすい色使いや、目をひく色づかいというものが存在する。[石田 97,]

地の色と文字色については、明度差が大きいもののほうが読みやすいという実験がある。そして、目をひきつけやすい色は、有彩色で暖色系で彩度の高いものであると言われている。[千々岩 78] そして、同系色の中に補色の関係で色が入ってきたとき、同じような形が並ぶ中に別の形がはいってきたときなども、目をひきつける。

CafeOBJ で仕様を記述する場合、おおまかにいって形をあらわす signature 部とアルゴリズムをあらわす axiom 部が存在する。この研究で製作したシステムは、あまり CafeOBJ に馴染みのない人間を対象としているために、例として出した CafeOBJ で書かれた仕様は短い。そのため、仕様の書き方は、signature 部と axiom 部にわけることにした。背景色を調整することで、モジュールの中において、signature と axiom がわかれていることをはっきりと示した。そのことにより、学習者が仕様の見たい部分を探すときの戸惑いを減らすことを狙った。背景色を変化はさせているが、強調すべき部分に対する色を残すために、色相的には、あまり離れていない色を使った。

検証の部分においては、補題ごとに区切って、すこしずつ見せている。その時、どこから先を、新しく提示したかを示すために、誘目性の高い色を用いた区切りを作っている。このシステムでは、大体、表題などを緑系の色であらわしている。区切りはオレンジ色になっているので、ラインが引かれているということのみならず、補色の関係から誘目性が高くなり、テキストのなかで目を引くことになる。

そして、検証の部分において、仮定と定義では彩度を変化させた。仮定も定義も、検証中ではどちらもポイントとなる場所である。重要であるということを、文字の大きさで示し、意味の違いを、彩度で示した。

第 5 章

CafeOBJ のラーニングシステムの構成

ここでは、実際に作った CafeOBJ のシステムの特色を解説する。

5.1 全体的な構成

システムの構成は、大まかに言って、CafeOBJ に関する解説であるところの導入部、実際に実際に仕様をかくということに関する学習を行う部、そして書いた仕様を CafeOBJ のシステムを使って検証する部の 3 つにわけることができる。

図 5.1:

図にある通り、最初に、いわゆる目次の形で、学習者にこれからの学習内容を示す。くりかえし利用する

ことが前提となっているので、そこからある程度学習内容に対してのリンクがある。

その中で、導入の部分は学習者の好みで、どれを選んでもいいようにしてある。しかし、仕様の記述に関する解説の部分や、検証に関する部分は、流れこそ示してあるが、すべていわゆる目次となっているわけではない。

類似性を強調することで、理解の手がかりを与えようといった意図があるため、単体で見られては困る部位もあるためだ。そのため、流れを示す最小限のものが、記述と検証に関しては、もくじになっている。

ことで、学習者にこれからの学習内容を把握させ、自分が学習しているのはどのあたりかを知らせることとする。

最初から学習する場合を考える。その場合は、まずはじめにいわゆる目次と簡単な内容説明から、学習内容のキーワードを得る。

その後、まず、導入部で仕様記述などに関する簡単な知識を提示する。あまり深い説明ではなく、簡単に特徴を解説する程度となっている。

それから、仕様記述に関する学習に入る。もくじで表題となっていた部分の学習の後、そのままその記述を用いた検証のやり方の学習にはいる。検証の部分が終わると、再び別の例題の記述に戻り、同じことをくりかえす。

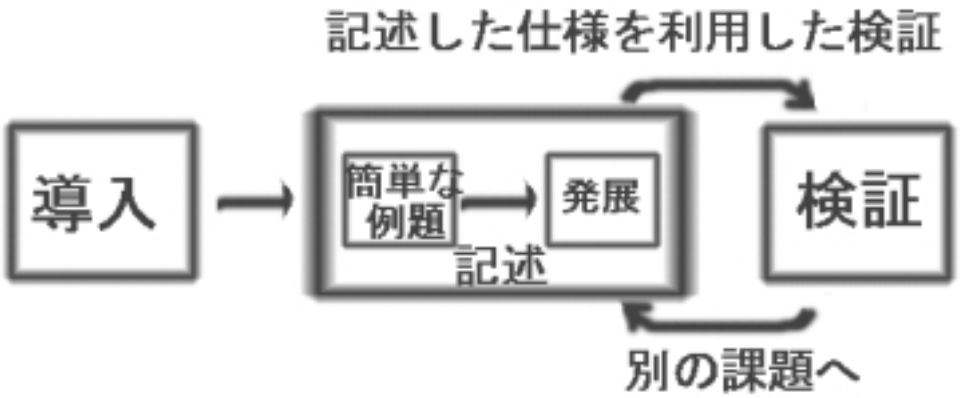

図 5.2:

これは、例題に対する理解が新しいうちに、それを利用することで、繰り返しによる学習の効果と、達成感を与えることを狙っているためである。

デザイン的には、地の色を薄いグレーにし、解説の文字の色を黒にしてある。地の色と、表示される文章の明度差をおおきく取ることで、コントラスト感をあげて、内容の見やすさを狙うために、色を選択した。

表題を始めとする、強調するべき部位は、強調の度合いの高いところから、濃い色を使い、全体的には緑系の色でまとめてある。

全体の色彩を統一することで、意味の強調のしかたのランク付けを行った。

強調する部分の色は、明度差こそあるが、色相的に近いものを用いてある。その理由は、内容的に強調する場所ではなく、次に学習者に目で追って欲しい場所を示すのに、色相的には補色に近い、最も誘目性の高い色を使うためである。

図 5.3:

5.2 導入

導入として、簡単な形式手法、形式仕様、CafeOBJ の特徴といった説明を用意した。最初に見る部分でもあるので、一度に学習者に提供する情報の量は少なく抑えてある。

記述や検証の部分とは違い、ハイパーテキスト構造をしていること以外、とりたてて特徴はない。

平行して同時に二つの解説を見るかといったことや、どの解説を最初に見るかなどのことは、すべて学習者側にまかせてある。

ここは、学習者に何かを理解させるというよりは、ある知識を提示するといった場所としてあるためである。記述や検証のように、新しいスキーマの形成を狙っている場所ではないため、学習のはじめとしては、流し読みされてもかまわないと考えている。

そのため、実際の記述方法を学んでいる時などに、何度も見ると考え、内容以外の場所、特に表示が遅いとか、一文が長すぎるなどのところで、ストレスを感じないようにといったことに留意した。

とくに、モデルに関する解説は、階層化し、上のほうではきついモデルと、ゆるいモデルに関する簡単な説明にとどめ、さらにリンクをたどることで、より深いまたは初歩的な知識を得られるようにしてある。

この部分では、今から学習することに対しての簡単な知識を得、知的好奇心を抑制しない程度に内容に興味を抱かせることを目標としている。

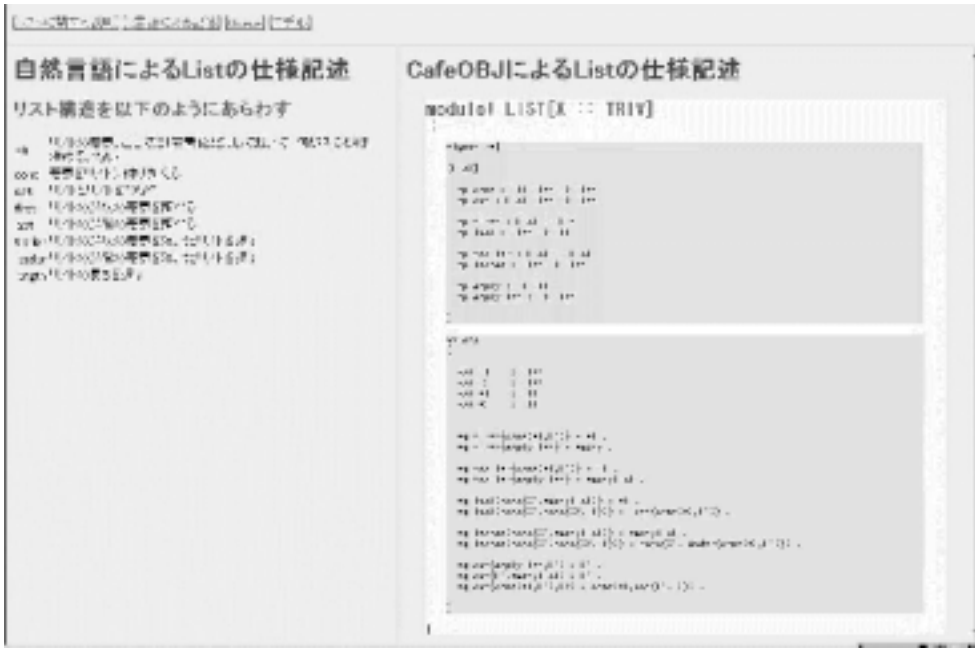

図 5.4:

5.3 記述

システムは、形式手法になじみのない人間を対象としている。そのため、例題には、ごく基本的なデータ構造 [Pate 88,] [西原 93,] を用いた。

学習の範囲は、主にきついモデルを定義する仕様とし、一題だけ、ゆるいモデルを定義する仕様を解説する。

画面を左右二つに分割し、仕様の書き方と、その書き方を理解するためのヒントを同時に並べて表示する。課題となるほうを右、それに関する説明を左と、常に位置を変えずに提示することで、新しい単元にはいった時の、学習者の視点のふらつきを減らすことを狙った。

さらに、完全に中央で二分割すると、学習者に与える印象が散漫になってしまうため、分割の割合を変え、学習のメインとなっている場所、つまり右側の印象を強くした。

CafeOBJ の表現的な面では、module、signature、axioms と、それぞれを記述した部分に対応する色を決め、モジュールの構成をはっきりを示した。そうすることで、記述の基本的なかたちを印象付けるとともに、どこに何が書いてあるかをわかりやすくした。module の記述は、引用のかたちとし、本文より背景色が明るく、文字の大きさも小さくなっている。

まず最初に、簡単な CafeOBJ の例題と、自然言語によるその説明を示す。自然言語による説明は、図などを用いて直感的理解をめざす。さらに、例題によっては、他のプログラミング言語による記述例を用いて [河西 92,]、理解の手がかりを増やし、例題そのものがわからないといった場合を極力避ける。

つぎに、先に説明した簡単な CafeOBJ の例題を拡張して、別のもっと複雑な仕様を記述する。これは、

理解の手がかりとして、問題の類似性を強調することを目指している。良く似た例を、間近に用いることで、白紙の状態で例題を与えられた場合の戸惑いを減らすことを狙っている。

たとえば、自然数と整数や、Queue と Deque などでそういった解説の方法を取っている。簡単なほうの例題と同じように、左側に自然言語と図による解説、右側に CafeOBJ による仕様を表示し、さらに新しいウィンドウを呼び、もとになった例題の CafeOBJ による記述を表示する。さらに、記述した仕様の中の共通部分の背景色を変えることで、記述した仕様の中の共通点を強調してある。

例題によっては、拡張したと言うよりは、利用している場合もある。List の解説のあと、それを利用し、簡単なファイルエディタの仕様を記述する。この場合、List の構造は輸入することになるので、色を変えて見せたりはしない。

輸入に関して触れた後、再び前に提示した仕様に戻り、データ構造を利用した操作を記述させる。これは、いくつかの実行の例題を示し、自然言語と図で与えた課題を学習者に実際に記述させることで、学習内容の定着を狙っている。

List 構造の説明を例にする。まず、CafeOBJ による記述を、右側に、それに関する解説を左側に表示する。

CafeOBJ による記述は以下のようになる。

```
module! LIST[X :: TRIV] {

signature{

[List]

  op cons : Elt List -> List
  op cat : List List -> List

  op first : List -> Elt
  op last : List -> Elt

  op trailer : List -> List
  op leader : List -> List

  op empty : -> Elt
  op emptylist : -> List

}
axioms
{
```

```
  var L1    : List
  var L2    : List
  var X1    : Elt
  var X2    : Elt


  eq first(cons(X1,L1)) = X1 .
  eq first(emptylist) = empty .

  eq trailer(cons(X1,L1)) = L1 .
  eq trailer(emptylist) = emptylist .

  eq last(cons(X1,emptylist)) = X1 .
  eq last(cons(X1,cons(X2,L1))) = last(cons(X2,L1)) .

  eq leader(cons(X1,emptylist)) = emptylist .
  eq leader(cons(X1,cons(X2,L1))) = cons(X1,leader(cons(X2,L1))) .

  eq cat(emptylist,L1) = L1 .
  eq cat(L1,emptylist) = L1 .
  eq cat(cons(X1,L1),L2) = cons(X1,cat(L1,L2)) .

}
}
```

リストの構造の説明は以下のようになる。

リスト構造を以下のようにあらわす
`Elt` リストの要素。ここでは「要素」とだけしておいて、何が入るかは決めていない
`cons` 要素をリストに付け加える
`cat` リストとリストをつなぐ
`first` リストの最初の要素を調べる
`last` リストの最後の要素を調べる
`trailer` リストの最初の要素を除いたリストを返す
`leader` リストの最後の要素を除いたリストを返す
`length` リストの長さを返す

CafeOBJ によるこのようなリストの記述を例にする。それに対する、自然言語による仕様は、ごく簡単な記述にとどめる。説明されている文は短いが、それぞれの関数をえらぶことで、どういったどうさをするかの図が表示されることになる。リストの構造は、自然言語による長い説明を行うことよりも、図による直感的理解を試みた方がいいと思われ、そういう構造にした。さらに、関数を選択することで図を見せるというやりかたは、学習者側の能動的な動きを増やすことで、注意力が散漫になることを防ぐことを目的としている。

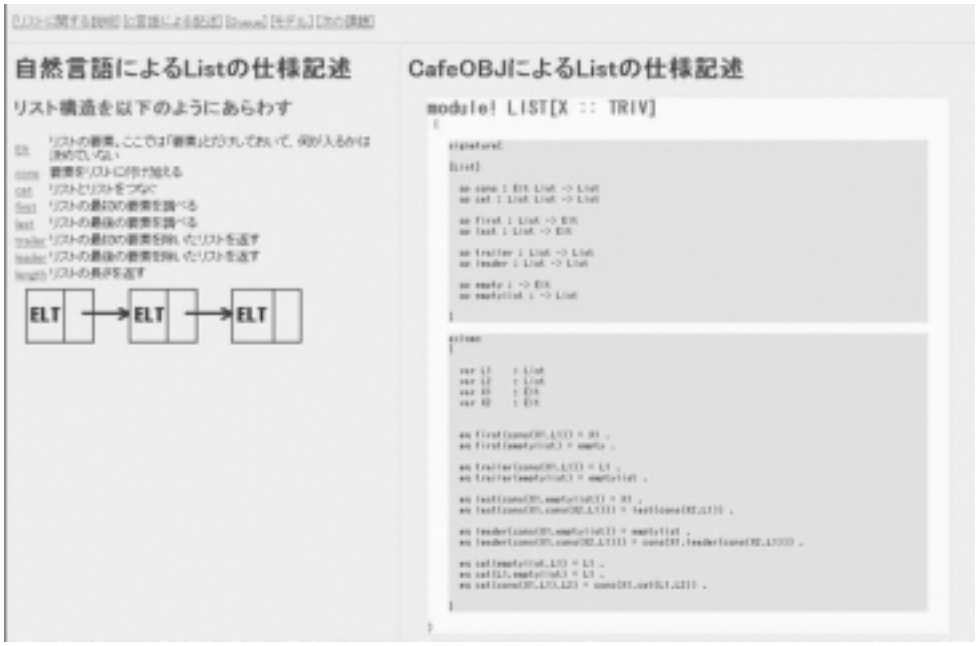
自然言語によるListの仕様記述
リスト構造を以下のようにあらわす
ELT
ELT
ELT
CafeOBJによるListの仕様記述
module! LIST[X :: TRIV]

図 5.5:

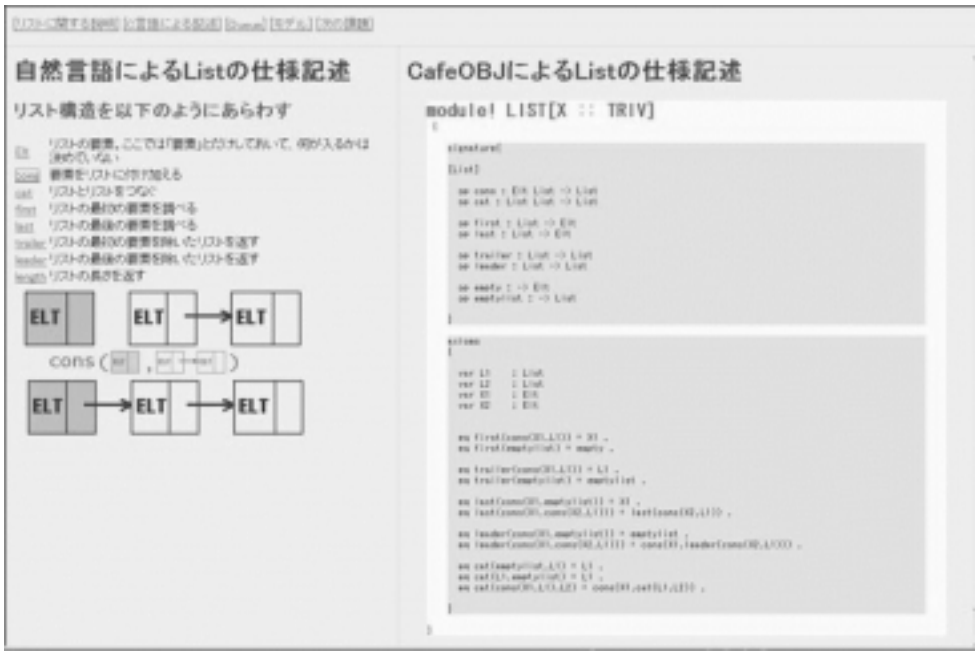
自然言語によるListの仕様記述
リスト構造を以下のようにあらわす
ELT
ELT
ELT
cons
ELT
ELT
ELT
CafeOBJによるListの仕様記述
module! LIST[X :: TRIV]

図 5.6:

5.4 検証

検証の課題のもとになる仕様は、CafeOBJ の記述の部分で示したものを利用している。これは、学習者が獲得した知識を利用することで達成感を持ち、肯定的なメタ認知を持つよう促している。知識を獲得しきっていない場合であっても、目標を示すことで、学習を継続する動機となりうることが期待される。

図 5.7:

検証の過程で、ポイントになる部分を見出しとして扱い、検証の流れを追いやすいようにした。これは、熟練者の視点の持ち方というスキーマを提示することで、学習者の CafeOBJ を用いた検証に対するスキーマの形成を促すことを狙っている。具体的には、等式を仮定したりする場所や、オペレーターの定義、実際にはコメントとして書き入れるべき部分を、それぞれ違った大きさや、違った彩度で表してある。特に、等式に関しては、それが仮定であるのか定義であるのかは、文脈を追わないとわからない。一目で全体像を掴めるようにしたいため、表示を別のものとした。

一度に、検証の過程を長く見せるのは、読む側の注意力が散漫になりやすい。とくに、あまり良くわかっていない場合においては、同じものの羅列に見えてしまったりするため、知的好奇心が抑制されることが考えられる。そのため、ある程度のかたまり、実際には補題ごとにくぎって、少しずつ見せるようにしている。

先に目を通したはずの部分を見るには、さかのぼって見るだけでいい。しかし、より先の部分を見るには、多少の動作が必要になることで、区切りをはっきりと示し、そこまでの確実な理解を促す構造になっている。

少しずつ見せた部分を確実に読んでいき、次の部分を見るかどうかということを自分で選ぶことにより、達成感を持たせ、継続の動機を持ちつづけさせることを狙う。さらに、ある程度の区切りを示すことで、検

証の構造を示し、スキーマの形成を促すことを狙っている。

検証の過程を理解するためには、書き換えの過程を見ることも手がかりになる。しかし、書き換えの過程をそのまま書いてしまうと、どうしても全体が煩雑になってしまい、流れがつかみにくくなってしまう。そのまて、書き換えの過程は、別の小さなウィンドウを呼ぶことで、見たいときに見るようにしてある。

対象とする人間は、あまり仕様記述になじみのない人間と限定してある。とはいえ、理解が早い、または逆に、先に進んだ学習者が後戻りして、部分を見たいということもありうるだろうし、一旦中断した学習の続きを行いたいといった場合も考えられる。そのためもあり、書き換え過程のウィンドウを呼ぶかどうかの判断は、学習者がわに任せてある。

5.4.1 実際の例題の自然言語による記述

実際に用いた例題について、方針をどのようにあてはめたかを説明する。

自然数における 2*(1+2+…+n) = n(n+1) の証明がひとつの例題となっている。この例題を択んだ理由は、例題そのものは基本的な帰納法の問題であり、一般的と思われるからである。CafeOBJ を用いて検証することの良さは少ないかもしれないが、例題そのものは馴染んでいるとは言いづらくとも、高校生程度の数学の課題であり、新しくスキーマを形成する場合の手がかりが多い。

```
2*(1+2+…+n) = n(n+1) を証明する
n=0 の時
(左辺)=2*0=0

(右辺)=2*0(0+1)=0

(左辺)=(右辺)
n=k の時 2*(1+2+…+k)=k(k+1) が成立するとする
n = k+1 の時を考える
(左辺)=2*(1+2+…+k+1)
       =2*(1+2+…+k)+2(k+1)

(右辺)=(k+1)(k+1+1)
       =k^2+3k+2
       =k(k+1)+2(k+1)

(左辺)=(右辺) が成立
よって 2*(1+2+…+n)=2n(n+1) は成立する
```

以上のような証明から、以下の部分を表題として抜き出す。

n=0 の時

n=k の時 2*(1+2+…+k)=k(k+1) が成立するとする
n = k+1 の時を考える

よって 2*(1+2+…+n)=2n(n+1) は成立する

実際の検証の部分から強調して、証明の流れを示すためである。これと対比して、CafeOBJ による検証を表示する。

その他、基本的なデータ構造を利用したものについては、直感的理解を求めるために、図を多用してある。記述の章で例にした List 構造の他に、Queue、Deque などについてもおなじことを行っている。つまり、上部には簡単な自然言語による説明、そして、下部には選んだ関数についての図を表示するスペースをもうけた。

5.4.2 実際の例題の CafeOBJ による記述

この例題に関して言えば、自然言語で証明した場合に比べ、どうしても CafeOBJ を用いて証明した場合の方が全体的に記述が長くなってしまう。

なじみがない方の記述が長いというのは、学習者の意欲に対して、あまりいい働きかけはしないように思われる。新しい情報を一度に多く与えると、学習者側が混乱しやすいということも考え合わせ、検証の過程を少しずつ表示することとした。

2*(1+2+…+n)=n(n+1) の証明
この証明で用いた自然数の仕様
sum(n) = 1 + 2 + ...+ n の定義

```
op sum : Nat -> Nat .
eq sum(0) = 0 .
eq sum(s(N)) = s(N) +0 sum(N) .

--> test
-- reduce in % : sum(0)
0 : Nat
(0.000 sec for parse, 1 rewrites(0.000 sec), 1 matches)
-- reduce in % : sum(s(0))
s(0) : NzNat
(0.000 sec for parse, 3 rewrites(0.000 sec), 4 matches)
-- reduce in % : sum(s(s(s(s(0)))))
```

```
s(s(s(s(s(s(s(s(s(s(0)))))))))) : NzNat
(0.000 sec for parse, 19 rewrites(0.017 sec), 33 matches)
[trace]
```

補題1・2 * (1 + 2 +....+ n) = n * (n + 1) の証明
任意の Nat k を仮定する

```
op k : -> Nat .
```

k = 0 の時 (1 + 2 +....+ n) + (1 + 2 +....+ n) = n * (n + 1) が成立する

```
-- reduce in % : 0 + 0 == 0 * (0 + s(0))
true : Bool
(0.000 sec for parse, 4 rewrites(0.000 sec), 4 matches)
```

sum(k) + sum(k) = k * (k + s(0)) を仮定

帰納法の仮定より

```
-- reduce in % : sum(s(k)) + sum(s(k)) == s(k) * (s(k) + s(0))
true : Bool
(0.017 sec for parse, 23 rewrites(0.033 sec), 103 matches)
```

よって sum(k) + sum(k) = k * (k + s(0)) は成立する

```
[trace]
```

補題2・2 * i = i + i の証明
2 * i = i + i の証明

```
op i : -> Nat .
```

i = 0 の時 2 * i = i + i が成立する

```
-- reduce in % : s(s(0)) * 0 == 0 + 0
true : Bool
(0.000 sec for parse, 3 rewrites(0.00 sec), 7 matches)
```

2 * i = i + i を仮定する

```
eq s(s(0)) * i = i + i .
```

--> 帰納法の仮定より

```
-- reduce in % : s(s(0)) * s(i) == s(i) + s(i)
true : Bool
(0.000 sec for parse, 8 rewrites(0.033 sec), 45 matches)
```
よって、2 * i = i + i が成立する

```
eq s(s(0)) * I:Nat = I + I .
--> よって、s(s(0)) * sum(n) == sum(n) + sum(n)
-- reduce in % : s(s(0)) * sum(i) == sum(i) + sum(i)
true : Bool
(0.000 sec for parse, 10 rewrites(0.033 sec), 80 matches)
```

2 * (1 + 2 +....+ n) = n * (n + 1) の証明
任意の Nat n を仮定する
op n : -> Nat .
n = 0 の時 s(s(0)) * sum(0) = 0 * (0 + 1) が成立する

```
-- reduce in % : s(s(0)) * sum(0) == 0 * (0 + s(0))
true : Bool
(0.000 sec for parse, 6 rewrites(0.017 sec), 18 matches)
```

s(s(0)) * sum(n) = n * (n + 1) を仮定
```
eq s(s(0)) * sum(n) = n * (n + s(0)) .
```
-> 帰納法の仮定より

```
-- reduce in % : s(s(0)) * sum(s(n)) == s(n) * (s(n) + s(0))
true : Bool
(0.000 sec for parse, 18 rewrites(0.050 sec), 127 matches)
```
2 * (1 + 2 +....+ n) = n * (n + 1) は成立する

以上の中から、以下の、自然言語による証明において強調した部分と、同じような流れを示す部分にあたる場所を抜き出して強調する。文字の大きさ、色などを揃えることで、検証の流れにおいておなじ位置にあることをあらわしている。

sum(n) = 1 + 2 + ...+ n の定義

補題1・2 * (1 + 2 +....+ n) = n * (n + 1) の証明

補題2・2 * i = i + i の証明

2 * (1 + 2 +....+ n) = n * (n + 1) の証明

2 * (1 + 2 +....+ n) = n * (n + 1) は成立する

自然言語による証明における、計算の部分を、CafeOBJ の処理系上での実行画面と考える。ただ、完全に同じというわけではないので、実際の実行画面であることを示すため、文字の大きさの対比は同じだが、背景色を変えてある。ここで表示する実行画面は書き換えの過程をトレースしたものではない。その理由は、学習者に一度に多量の情報をあたえすぎることは、混乱をおこさせ、注視する部分がぼやけてしまい、学習意欲の低下につながる可能性があるからである。

平行して表示し、類似部分を強調することで、新しいスキーマを作る際の、てがかりを示してある。

一度に表示する部分は、強調した部分ごとに区切った。つまり、等式の証明、および、導入の最小の単位の部分で区切ってある。一度に学習者に情報を与えすぎることは混乱を招くが、全体が見通せないほど細かく区切ることもまた混乱を招くことになる。

そのため、補題ごとに区切ることにした。区切りごとに、学習者がえらんで先を表示することにしてある。そのため、先に進んだときに、どこを見ればいいのかわかりにくくなるという状態を防ぐため、誘目性の高い色の記号を用いて、前から表示していた場所と、新しく表示した場所を区切った。

第 6 章

評価

本研究では、形式手法に馴染みのない人間が代数仕様言語 CafeOBJ について学ぶということを前提としたラーニングシステムを製作した。ここでは、学習内容、学習者側に想定される動作環境、全体的なデザインについてそれぞれ評価する。

6.1 学習内容について

導入、記述、検証と大まかに内容を区切ってシステムを製作した。ここでは、それぞれの部分に対する評価を行う。

6.1.1 導入部

形式仕様になじみのない人間が、形式手法で仕様を記述してみるということを目標としているためもあり、例題の範囲をくぎり、なおかつ例題そのものが長くならないようにすることにある程度成功している。

ただし、形式手法になじみのない人間を相手にしているわりには、導入部分における解説が親切とは言いがたい。導入部分のわかりやすさも、学習者のメタ認知に影響を与えるため、もう少し内容的な充実が必要だと思われる。

ハイパーリンク構造の階層の上の部分は、何度も参照することを考え、ある意味、スキーマが形成されている人間相手の簡単なメモでもいいかと思われる。しかし、そのメモ的なものに対して、より詳しい解説を求めた場合のケアが十分ではない。

これは、学習者に否定的なメタ認知を与え、継続の動機をそぐことにもなりうる。

ただし、ハイパーリンク構造で起こりやすい、読んでいる側が現在の自分の位置に当たる場所がわからなくなってしまう、ロストウェイといわれる現象はおこらないだろう。だがこれは、解説部分の分量そのものがあまり多くないためであって、構造が良いためとは限らないだろう。

6.1.2 仕様記述について

形式手法にあまり馴染みのない人間を対象にしているので、例題は簡単なデータ構造を選んでいる。このことで、システム内の解説に不備があった場合でも、学習者が比較的参考書をさがしやすくなっている。

そして、モジュールの構造にたいして色を決めることにより、わかりやすく構成を表現することに成功した。結果として、先に進んだ学習者が、もう一度記述されたものを見たいと思ったときに、探すのが楽になっている。

色々な記述方法で例を示してあるが、実際のところ、学習者が利用しきれるかどうかは疑問が残る。方向性を変えた解説よりも、個々のたとえば、自然言語による解説を充実させ、階層構造によりもっと深い説明が得られるようにするべきなのではないかとも思われる。

CafeOBJ による記述に関する部分はともかく、自然言語による記述の部分などでは、インタラクティブ性を持たせても良かったかもしれない。ただ、このシステムは、年齢的には少なくとも子供を相手にしているわけではない。自然言語による記述の部分で、ちょっとした操作をして理解の手助けになるようにしたとしても、特にシステムのレスポンスの低下があった場合に、逆に継続の動機をそぐことになる可能性もあるため、必ずしも必要とはいえないだろう。

6.1.3 検証について

仕様を記述する部分で用いたものを利用して検証を行うことで、知識の獲得による途中経過を見せることに成功した。獲得した知識が利用できることを知らせることで、知的好奇心をおこし、学習の継続の動機にプラスの影響をあたえている。

検証の過程を見せる際に、総てを一度に見せてしまうわけではなく、少しずつ区切りながら見せることを行った。このことで、一度に学習者に与えられる情報が大きくなりすぎるのを防ぎ、学習意欲の低下を防止している。

さらに、先を見ることに必要な作業を行わせることで、漫然と長い文書を目で追うわけではなく、部分部分の中での確実な理解が要求されることを表現できた。

だが、検証に関する部分では、学習者が実際に自分で検証の組みたてを行う例題が不足しているため、システムに検証させる場合のヒントにはなるが、定着をはかることができない。

6.2 利用者に要求する環境について

題材を、ハイパーテキスト構造にするために、HTML とそれに対応するブラウザを用いている。結果として、WWW で配布することが可能になった。しかし、WWW の利点であるところの、クライアント側の様々な環境に適応するといったようなことは、考えていない。

このシステムでは、色やフォントの大きさなども重要な理解の手がかりになるよう考えている。そのため、クライアント側をある程度制限せざるを得ない。これは、WWW の利点を利用しきっていないとも言えるが、システムの性質上、やむをえないと思われる。

利用者がわの環境を限定せざるをえない主な理由としては、CSS の利用があげられる。さらに、CafeOBJ による仕様記述をはじめとする、改行位置を指定してあるものなどがあり、それを利用者側の設定にまかせるように書きかえるのは可能だが、見やすさの面から見てあまり望ましいことではない。

そのため、WWW の利点といわれている、クライアント側の様々な環境に対応できるといったことは、いかしきっていない。

WWW の利点といわれるものを生かすためには、どうしてもデザイン的な面でどうにもならないところが出てくる。

学習者に要求する環境は以下のようなものになる。

- 色が表示される環境で利用すること
- CSS に対応したブラウザで利用すること
- グラフィックデータが表示できること
- いくつかのブラウザを同時に立ち上げられること
- CafeOBJ のサーバーが利用できることが望ましい
- ある程度以上のフォントの大きさが揃っていること

色を用いて注視点や、モジュールの構造などをあらわしている以上、学習者に対してカラー環境を要求しなくてはならない。

HTML を適切に記述したいがために、CSS を用いているので、ブラウザを使うならば、CSS の表示に対応したものを使わざるをえない。

自然言語による説明の部分で、図を用いるためにグラフィックが表示される環境であることが必要である。

あたらしいウィンドウを呼んで、同時に内容を表示している部分があるため、ブラウザがひとつしか立ちあがらないようでは、困る。

テキストブラウザを用いて表示した場合、フレームを利用して平行して表示している部分が全く表示できなくなってしまう。さらに、図を用いて説明を行っている部分が、全く意味をなしていない。そのため、事実上、利用は不可能になってくる。

CSS 非対応のブラウザの場合には、ごく普通の HTML 文書として表示される。デザイン的な工夫は一切なしになってくる。ある程度制限される部分はあるが、次の、中途半端に CSS に対応しているブラウザよりは、わかりやすい表示となっている。

背景の色を設定することで、説明のまとまりを示しているが、ブラウザによっては別の表示のやりかたをされてしまい、まとまりを示しているように見えない。この場合の改善方法は、存在する。しかし、HTML のコードが内容に対して妥当ではなくなってくるために、そういった方法をとることはしなかった。

以上のことから、WWW を用いることでもたらされる自由度は、多少制限されている。だが、それぞれ理由があってのことであり、現在の状況でならばさほど厳しい制約とはいえないため、問題ないだろうと思われる。

6.3 デザインについて

背景色の調整で、解説部分と仕様記述部分をはっきりと分離させることに成功した。解説部分と、実際の仕様記述がまざりあってしまうことからの、実際にどう書くかといったことに対する学習者の戸惑いを減らすことに貢献しているだろう。

背景色と文字色の関係、レイアウトなどで、多少は見やすくなっているが、Web において長い文章とは根本的に見づらい。いくつかの予測できる原因は存在するが、確実に言えることは、反射を利用しているわけではなく、光を発してものを見せている、ディスプレを利用していることである。いくら色や輝度をいじってみたところで、内容に集中して画面に向かうと、目の疲労が著しい。

当然、ある程度以上の長さの文書を集中して読むことを要求するラーニングシステムにとって、これは問題点である。

文書構造を、今現在必要と思われる以上にくわしくマークアップしてあるため、結果的に色彩的な変更は容易で、なおかつ、ソースそのものの再利用も容易になっている。実際のシステム内では作られていないが、仕様記述の特定の部分だけを検索して抜き出すようなことも可能な作りとなっている。

ただ、ディスプレイでテキストを見た場合の根本的な見づらさに関しては解決されていない。色彩を調整することで、ある程度目が疲れにくい、見やすいテキストにすることはできる。しかし、ディスプレイ上で長い文章を見た場合の疲れ方は、同じ量の紙に印刷したテキストに比べてはるかに大きい。

その原因として以下のようなことが考えられる

- 文字の表示方法の違い
- マージンの少なさ
- 一度に目に出来る情報の少なさ

マージンの少なさも、目に出来る情報の少なさも、もとを辿れば、文字の表示方法の違いにいきあたる。ディスプレイでは、紙に印刷するほど、小さな文字で、マージンをあけて内容を表示することが出来ない。

そのため、学習者が一度に目に出来る情報は、紙媒体を用いたものよりも少なくなってしまう。そのことにくわえて、発光することで文字を表示しているディスプレイでは、どうしても目が疲れてしまう。

確かに、学習者に対して、一度に多量の情報をあたえることは、混乱を招く。しかし、内容がある程度高度になってくると、どうしても、ある程度以上の内容を一度に与えざるを得ない。

やり方としては、最初からプリントアウトして学習者が利用するだろう部分を想定しておくことが考えられる。軽く情報を与える部分をブラウザで見るように作っておき、特に理解を促す部分はプリントアウトして利用するような資料を用意しておくことが望ましいだろう。

デザインそのものについては、センスのいい人間ならば簡単に作れる、または、もっと見やすいものができるということが言える。だが、この研究においては、背景色の決め方や、見せる分量などを、色々な根拠を用いて定めている。そのため、センスの問題ではなく、完全にとはいわないが、構造が再現できるはずである。そのことは、有用であるといえるだろう。さらに、CafeOBJ に特化した CSS を製作してあるため、ラーニングシステムに限らず、CafeOBJ による仕様記述を HTML 化して見せることを試みたときに有用だろう。

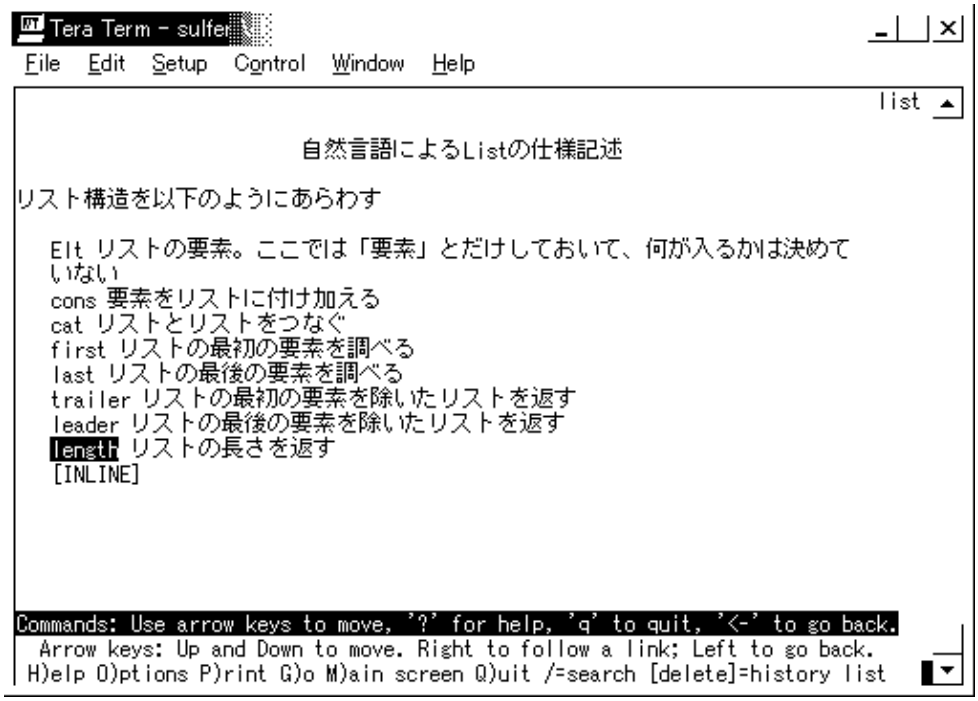
Tera Term - sulfer
File Edit Setup Control Window Help
list
自然言語によるListの仕様記述
リスト構造を以下のようにあらわす
Elt リストの要素。ここでは「要素」とだけしておいて、何が入るかは決めて
いない
cons 要素をリストに付け加える
cat リストとリストをつなぐ
first リストの最初の要素を調べる
last リストの最後の要素を調べる
trailer リストの最初の要素を除いたリストを返す
leader リストの最後の要素を除いたリストを返す
length リストの長さを返す
[INLINE]
Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list

図 6.1:

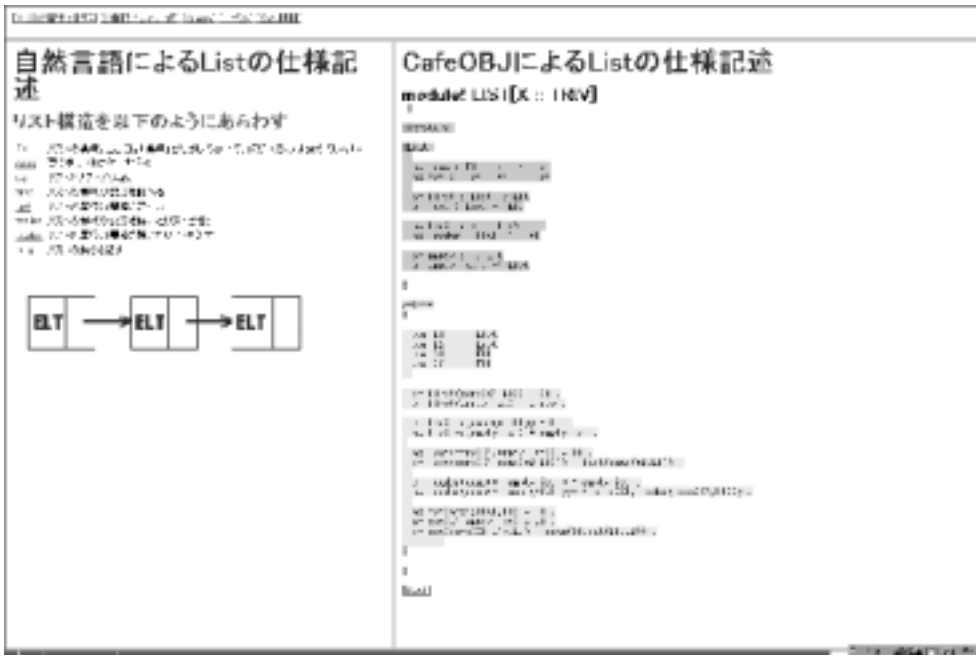
自然言語によるListの仕様記述
リスト構造を以下のようにあらわす
ELT
ELT
ELT
CafeOBJによるListの仕様記述

図 6.2:

第 7 章

今後の課題

今後の課題としては以下のようなものが考えられる。

- 特に検証部分の例題の充実
- 学習範囲の拡大
- 導入部分の解説の充実
- HTML ソースの利用

例題そのものの数が多いとはいえない。だが、特に検証部分の例題が不足している。記述の部分では、学習者に対して、実際に記述し、実行してみるべき例題が示されているが、検証の部分ではそういったものはない。例題を提示するだけでは、学習者の理解の定着のためのリハーサルにはなりにくい。そのため、学習者自身に記述させるものも含めた、特に検証部分における新しい例題が必要であると思われる。

今回、形式手法に馴染みのない人間を対象としているためもあり、学習の範囲を、きついモデルの記述にほぼ限ってある。CafeOBJ を対象としたラーニングシステムであるにも関わらず、振舞いモデルを定義する仕様に関する例題が存在していない。CafeOBJ の解説で、動的な振舞いを記述することが出来ると書いてあるにもかかわらず、例題にそういったものがない。以上のことから、学習範囲の拡大が必要であると思われる。

導入部分が、今のままでは、かなり学習者個人の学習動機と、知識にたよっている。解説の充実により、学習者の疑問に答える作りをめざし、知的好奇心を持たせるようにしむけることが望まれる。

具体的には、解説内容の増量と、それにともなった、構造の見なおしが必要だろう。

将来的に、例えば学習者が望んだ部分、例えば、等式の部分だけを仕様から抜き出して表示することも可能なように作ってある。デザイン的なやり方で、情報が見つけやすいようにはしてあるので、どうしても必要というわけではないので実装は行わなかった。だが、理解の手がかりとして使えるものではあるので、今後の課題として、あげておいた。

第 8 章

Appendix

8.1 CSS

HTML は文書構造を示すための言語であって、ブラウザで表示した場合のレイアウトを規定するものではない。従来は、レイアウトに関しては、それぞれのブラウザに任せられていた。そのため、レイアウトに関して HTML を記述する側が操作を行いたいと思った場合は、その場限りの実装をせざるをえなくなり、文書構造を示すという HTML の役割がぼやけてしまっていた。そして、もともとレイアウト用の言語でないものにレイアウトをさせようとするため、できることにも限界があった。

CSS は Cascading Style Sheet の略であり、HTML で記述された文書の見栄えを整えるためのものである。新しいクラスを指定することにより、より細かく HTML タグの意味を分けることもできる。

CSS を用いることで、本来の HTML の役割であるところの、文書構造を適切にあらわすことができるようになった。さらに、見栄えとしては変わらなくても、CafeOBJ による仕様記述などの部分に、意味の違ったタグを設定してあるため、HTML のソースに対して、検索などの作業を施すことが可能になっている。

さらに、意味が一貫しているために、レイアウト的な変更も容易になる。

8.2 全体に対する設定

まずは、全体に対して、一般的な色彩や、表題の文字の大きなど共通する部分を設定した。強調する部分の文字の大きさや、色、明度などがまちまちであると、場所によってどれが重要かといったことをいちいち見極めなくてはならない。わざわざ、強調のしかたなどを変える必要はないと思われたため、全体的に、淡いグレーの背景に、強調部分は明度と大きさを変えた鈍い緑色で統一してある。さらに、リンク部分なども、特に場所によって設定を変える必要がないと思われるため、共通部分として設定した。

```
/* 背景色と文字色 */
body
{
  background: #eeeeee;
  color: black;
}

/* コーナータイトル */
H1
{
  font-size: x-large;
  color:#0c5a66;
}

/* これから書いたり証明したりするものの内容 */
h2
{ font-size:large; }

/* 整形済み。引用など */
pre
{
 margin-left: 2.0em;
 margin-right: 2.0em;
 margin-top: 1.0em;
 margin-bottom: 1.0em;
}

/* リンク色 */
A:link{ color: #009999}
A:visited{ color: #008080}
A:active{ color: red }

/* 文章 */
P
{
```

```
  display: block;
  white-space: normal;
  text-indent: 1.2em;
  margin: 0.0em;
}
```

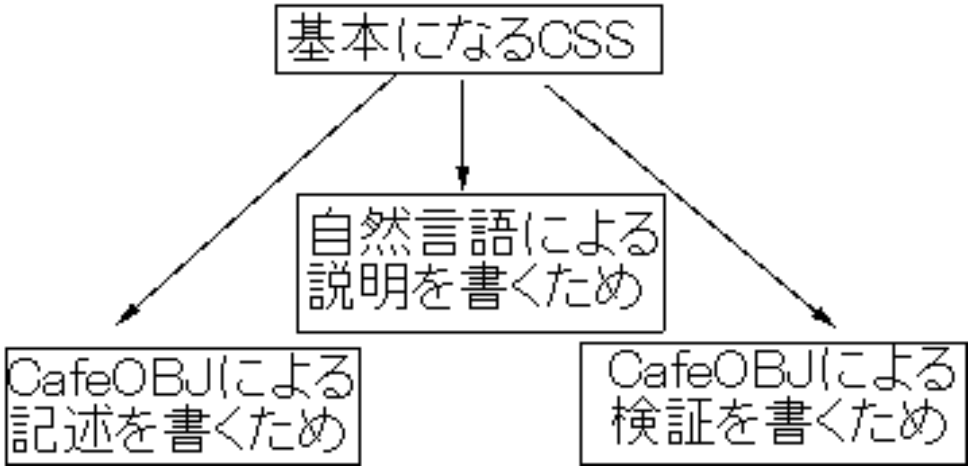

図 8.1:

8.3 自然言語による説明に対する設定

自然言語による説明に対する設定では、基本となる設定を輸入し、さらに画像を埋めこむための設定が追加されている。

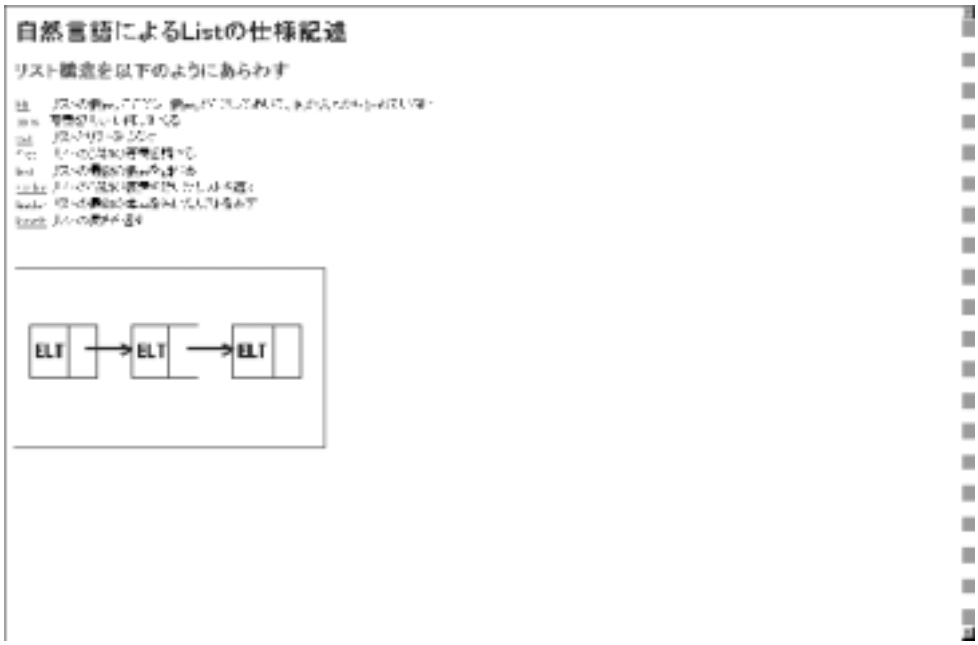

図 8.2:

```
H2
{ color: #2f7e88; }

H3
{
  color: #2f7e88; */
  font-weight: bold;
  margin-top: 0.0em;
  margin-bottom: 0.0em;
}

div.pict
{
  border-style: ridge;
  border-width: 2px;
```

```
  margin-top: 3em;
  height: 15em;
  width: 10em;
}
```

8.4 仕様記述に対する設定

CafeOBJ による仕様記述の場合には、基本となる設定を輸入し、以下のような設定を付け加える。

```
/* モジュールの背景 */
/* 本来なら div.* となって、改行の設定をしなくてはいけない */
/* white-space: pre; */
pre.mod
{ background: #fcfcfc; }

pre.signature
{ background: #ddddff; }

pre.axioms
{ background: #cfeedf;}
```

以上のような設定を付け加えることで、CafeOBJ による仕様記述において、module と signature と axioms の部分のかたまりを示すことができるようになっている。実際にこれを用いて記述した HTML は以下のようになる。

```
<html>
 <head>
 <title>List in CafeOBJ</title>
 <link rel="stylesheet" href="../css/des.css" type="text/css">
<META http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
</head>
<body>
<h1>CafeOBJ による List の仕様記述</h1>
<pre class=mod>
<h2>module! LIST[X :: TRIV]</h2> {
```

```
<pre class=signature>signature{

[List]

  op cons : Elt List -> List
  op cat : List List -> List

  op first : List -> Elt
  op last : List -> Elt

  op trailer : List -> List
  op leader : List -> List

  op empty : -> Elt
  op emptylist : -> List

}
</pre>
<pre class=axioms>
axioms
{

  var L1    : List
  var L2    : List
  var X1    : Elt
  var X2    : Elt


  eq first(cons(X1,L1)) = X1 .
  eq first(emptylist) = empty .

  eq trailer(cons(X1,L1)) = L1 .
  eq trailer(emptylist) = emptylist .

  eq last(cons(X1,emptylist)) = X1 .
```

```
  eq last(cons(X1,cons(X2,L1))) = last(cons(X2,L1)) .

  eq leader(cons(X1,emptylist)) = emptylist .
  eq leader(cons(X1,cons(X2,L1))) = cons(X1,leader(cons(X2,L1))) .

  eq cat(emptylist,L1) = L1 .
  eq cat(L1,emptylist) = L1 .
  eq cat(cons(X1,L1),L2) = cons(X1,cat(L1,L2)) .

}
</pre>
}

</pre>
<a href="file_index.html" target="_top">[NEXT]</a>
</body>
</html>
```

signature や axioms の部分が互いに別のタグでかこまれているために、その部分だけ抜き出すといった機械的処理をつけくわえることが容易である。CSS を利用しなかった場合、背景色の設定の違いといったような意味のない違いになってしまう。CSS を利用することで、色の変更が容易になったのみならず、機械的処理にも向いた HTML が記述できるようになった。

8.5 検証に対する設定

CafeOBJ をもちいた検証を表示するための設定は、基本の設定を輸入し、情報を小出しにするためのしかけに関する設定、および、解説部分と CafeOBJ による記述を引用した部分、などの設定が行われている。さらに、定義と仮定に対する設定が行われている。定義と仮定に対する設定は、文字の大きさ的に変化はない。しかし、意味の違いをあらわすために、彩度を変化させた。そして、その副産物として、機械的処理を行いたいと思った場合に利用できるようになっている。この設定がある為に、CafeOBJ で書かれたものを、そのまま機械的な処理で HTML 化することはできなくなった。だが、それは題材の性質上仕方のないことだろう。

```
/* 成立したもの */
H4
{
 color: #2f7e88;
```

```
 margin-left: 2.0em;
 margin-bottom:0.2em;
 font-size: normal;
}

/* ここから下が新しい表示部分 */
hr.sign{ color: #f93; }

/* red */
pre{ background: white; }

/* 補題 */
h3{ color: #2f7e88; }

/* eq */
div.eq
{
  font-weight: bold;
  color: #1f6e78;
  margin-left: 2.0em;
}

/* 仮定 */
div.if
{
  font-weight: bold;
  color: #2f7e88;
  margin-left: 2.0em;
}
```

このCSSを用いて、検証部分の説明を記述すると、以下のようになる。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
 <head>
 <title>Nat induction</title>
```

```
 <link rel="stylesheet" href="../css/induction.css" type="text/css">
 <META http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
</head>
<body>
<script language="JavaScript">
<!--
 function opennew(loc)
 {
   var wnd="opennew";
   var width='600';
   var height='200';
   var myWindow=window.open(loc,wnd,'resizable=yes,scrollbars=yes,
       status=0,width='+width+',height='+height);
   if (myWindow.focus!=null) {
          myWindow.focus();
      }
    }
-->
</script>
<h1><自然数の仕様を用いた証明</h1>
<h2>2*(1+2+…+n)=n(n+1) の証明</h2>
<a href="nat.html" target="left">この証明で用いた自然数の仕様</a>
<h3>sum(n) = 1 + 2 + .....+ n の定義</h3>
<div class=eq>
op sum : Nat -> Nat .<br>
eq sum(0) = 0 .<br>
eq sum(s(N)) = s(N) +0 sum(N) .<br>
</div>
<pre>
--> test
-- reduce in % : sum(0)
0 : Nat
(0.000 sec for parse, 1 rewrites(0.000 sec), 1 matches)
-- reduce in % : sum(s(0))
s(0) : NzNat
```

```
(0.000 sec for parse, 3 rewrites(0.000 sec), 4 matches)
-- reduce in % : sum(s(s(s(s(0)))))
s(s(s(s(s(s(s(s(s(s(0)))))))))) : NzNat
(0.000 sec for parse, 19 rewrites(0.017 sec), 33 matches)
</pre>
<a href="javascript:opennew('induction_t1.html')">[trace]</a>


<h3>補題1・2 * (1 + 2 +....+ n) = n * (n + 1) の証明</h3>
<div class=if>任意の Nat k を仮定する<br>
op k : -> Nat .
</div>
<h4>k = 0 の時 (1 + 2 +....+ n) + (1 + 2 +....+ n) = n * (n + 1) が成立する</h4>
<pre>
-- reduce in % : 0 + 0 == 0 * (0 + s(0))
true : Bool
(0.000 sec for parse, 4 rewrites(0.000 sec), 4 matches)
</pre>
<div class=if>
sum(k) + sum(k) = k * (k + s(0)) を仮定<br>
帰納法の仮定より
</div>
<pre>
-- reduce in % : sum(s(k)) + sum(s(k)) == s(k) * (s(k) + s(0))
true : Bool
(0.017 sec for parse, 23 rewrites(0.033 sec), 103 matches)
</pre>
<h4>よって sum(k) + sum(k) = k * (k + s(0)) は成立する</h4>
<a href="javascript:opennew('induction_t2.html')">[trace]</a>

<h3>補題2・2 * i = i + i の証明</h3>
<h4>2 * i = i + i の証明<br>
op i : -> Nat .
</h4>
<h4>i = 0 の時 2 * i = i + i が成立する</h4>
```

```
<pre>
-- reduce in % : s(s(0)) * 0 == 0 + 0
true : Bool
(0.000 sec for parse, 3 rewrites(0.00 sec), 7 matches)
</pre>
<div class=if>2 * i = i + i を仮定する<br>
eq s(s(0)) * i = i + i .
</div>
<pre>
--> 帰納法の仮定より
-- reduce in % : s(s(0)) * s(i) == s(i) + s(i)
true : Bool
(0.000 sec for parse, 8 rewrites(0.033 sec), 45 matches)
</pre>
<h4>
よって、2 * i = i + i が成立する<br>
</h4>
<div class=eq>
eq s(s(0)) * I:Nat = I + I .
</eq>
<pre>
--> よって、s(s(0)) * sum(n) == sum(n) + sum(n)
-- reduce in % : s(s(0)) * sum(i) == sum(i) + sum(i)
true : Bool
(0.000 sec for parse, 10 rewrites(0.033 sec), 80 matches)
</pre>

<hr class=sign>
<a name="next"></a>
<h3>2 * (1 + 2 +....+ n) = n * (n + 1) の証明</h3></a>
<div class=if>
 任意の Nat n を仮定する<br>
op n : -> Nat .
</div>
<h4>
```

```
 n = 0 の時 s(s(0)) * sum(0) = 0 * (0 + 1) が成立する
</h4>
<pre>
-- reduce in % : s(s(0)) * sum(0) == 0 * (0 + s(0))
true : Bool
(0.000 sec for parse, 6 rewrites(0.017 sec), 18 matches)
</pre>
<div class=if>
s(s(0)) *  sum(n) = n * (n + 1) を仮定<br>
eq s(s(0)) *  sum(n) = n * (n + s(0)) .
</div>
<pre>
-> 帰納法の仮定より
-- reduce in % : s(s(0)) * sum(s(n)) == s(n) * (s(n) + s(0))
true : Bool
(0.000 sec for parse, 18 rewrites(0.050 sec), 127 matches)
</pre>
<h2>
2 * (1 + 2 +....+ n) = n * (n + 1) は成立する
</h2>

<a href="cil_f.html">[実行画面]</a>

</body>
</html>
```

図 8.3:

図 8.4:

参考文献

[福島 93] 福島学，浮貝雅裕，菅原研次，城戸健一. プログラミング演習のためのハイパテキスト型教材の実装, 情報処理学会論文誌, 1Vol.34,No.6,pp.1246–1257(1993).

[芳賀 93] 芳賀博英, 小嶋弘行. ハイパーメディアを用いた実習支援機能付きプログラミング教育用 CAI システムの開発, 情報処理学会論文誌, Vol.34,No.11,pp.2302-2311(1993).

[溝口 95] 溝口理一郎, 知的教育システム, 情報処理, Vol.36,No.2,pp.177-186(1995).

[大林 98] 大林史明:インターネットを用いた知的教育システムの認知心理学に基づく設計と実験評価 URL:http://hydro.energy.kyoto-u.ac.jp/Lab/ronbun/obayashi.html .

[中川 96] 中川中, 谷津弘一, 本間毅寛:CafeOBJ への誘い URL:http://www.ipa.go.jp/STC/CafeP/cafeproject.html .

[坂元 91] 坂元昴, 教育工学, 放送大学教材, 放送大学教育振興会 (1991).

[Răzvan 98] Răzvan.Diaconescu,Kokichi.Futatsugi, CafeOBJ Report,World Scientific(1998).

[Pate 88] Pete,T.Hugh,R.Judy,M.Abstract Data Types,Computing science series,Oxford University Press(1988).

[Joseph 98] Joseph,Goguen,at el.LINKS for a Hidden World URL:http://www.cs.ucsd.edu/groups/tatami/ .

[千々岩 78] 千々岩英影, カラーコントラスト感スケール作成の試み, 日本色彩学予稿集.

[石田 97] 石田恭嗣, いしだみどり, カラーコーディネーター合格ハンドブック, 明日香出版 (1997).

[千々岩 83] 千々岩英影, 色彩学, 福村出版 (1983).

[西原 93] 西原清一, データ構造, 新コンピュータサイエンス講座, オーム社 (1993).

[河西 92] 河西朝雄,C 言語によるはじめてのアルゴリズム入門, 技術評論社 (1992).

[御領 93] 御領謙, 菊地正, 江草浩幸, 最新認知心理学への招待–心の働きとしくみを探る–, 新心理学ライブラリ 7, サイエンス社 (1993).

[James 91] James Rumbaugh, et al,Object-Oriented Modeling and Design,PrinticeHall(1991)(邦訳:羽生田栄一:オブジェクト指向方法論 OMT, トッパン (1992)).