修 士 論 文

区間遷移システム:
複数の環境における
プログラムの意味の集約

令和四年度 修了
三重大学大学院　工学研究科
博士前期課程　情報工学専攻
コンピュータソフトウェア研究室

近藤 流司

概要

様々な計算機 やプログラミング言語が日々開発されている．そのため古くなった計算機を新調して移植したりプログラミング言語を変えてこれまでのプログラムを作り変えたりすることがある．そこで，プログラムが正しく動作することやプログラムの最適化，プログラム間の等価性を評価する必要がある．したがって，プログラムを解析することは重要である．

遷移システムは各ブロックにおける変数の値を状態とし，変数の値の変化を遷移とする状態遷移を用いて表現することでプログラムを解析する手法であり，トランスパイラの等価性の保証などに用いられる．プログラム解析ではプログラムの流れを解析する制御フロー解析や各ブロックにおける変数の取り得る値の集合を解析するデータフロー解析等があり，プログラムの最適化に用いられる．これらは一つの決まった実行環境に基づいて，プログラムやプログラム変換に対して検証することは可能である．しかし，実行環境が増えるほどプログラム解析のための検証器の数は急増し，更に複数の実行環境やプログラミング言語の仕様を必要とするトランスパイラの検証器やプログラム間の検証器は膨大な数が必要となる．これはそれぞれ開発者にとっては開発すべき数が増えてしまうという点で，使用者にとっては使用している計算機やプログラミング言語の仕様の組み合わせに適した検証器を選択が必要な点で負担になる．そこで，複数の仕様を満たしながらプログラムを解析できればこれらの問題は解決できる．数値解析では計算機で実装された値や関数に対し，閉区間とその上の演算を与えることで関数の結果がどの程度の大きさの区間であるか調べることで関数の精度を考察する区間解析がある．

本研究では，区間解析の対象を関数からプログラムに拡張し遷移システムの各状態における変数の解釈として閉区間を取ることでプログラム実行後における変数の精度を調べるために，区間論理とその上の区間遷移システムを提案する．先行研究で用いられた論理を用いて各変数の値を閉区間として解釈して遷移システ ムを構成するだけでは複数の条件分岐を扱えず，各条件分岐においてその条件を満たすような仕様のみを表現する遷移システムを前提としてしまうため不十分である．これはプログラム中に現れる論理代数と遷移システムを構成するために必要な論理に現れる論理代数を同じものとして扱うことに起因する．そのため，それらの論理代数を分けた新たな論理である区間論理を定義し，その上に区間遷移システムの定義を与えることでこの問題を解決する．区間論理は計算機やアーキテクチャやプログラミング言語などの仕様の組みあわせの定式化であり，区間遷移システムはその上のソースコードの定式化である．

目次

第 1 章

はじめに

1.1 研究背景

サポートが終了した古い機械から新しい機械へ移管するときやプログラミング言語を新しいものに書き換えるとき，移管前後の動作が等価であることが求められる．2 つのプログラムの等価性は次の順序で示せる．

(1) 2 つのプログラムを定式化する
(2) 等価性を定義する
(3) 等価性を満たすことを証明する

ここで，等価性の与え方によってプログラムの定式化はソースコードの解析だけでは足りない場合がある．以下に例を示す．プログラム 1.1 は COBOL で書かれたプログラムであり，プログラム 1.2 は C 言語で書かれたプログラムである．

プログラム 1.1　COBOL のプログラム

```
1       IDENTIFICATION DIVISION.
2       PROGRAM-ID. ADD.
3       DATA DIVISION.
4       WORKING-STORAGE SECTION.
5       77 X PIC 9(1)V9(1) VALUE 0.1.
6       77 Y PIC 9(1)V9(1) VALUE 0.4.
7       PROCEDURE DIVISION.
8           IF X + Y < 0.5 THEN
9               COMPUTE X = (X + Y) / 0.5
10          ELSE
11              COMPUTE X = 1
12          END-IF.
13          STOP RUN.
```

プログラム 1.2　C 言語のプログラム

```
1 int main(void){
2   double x = 0.1;
3   double y = 0.4;
4   if (x + y < 0.5) {
5       x = (x + y) / 0.5;
```

```
6   }
7   else {
8       x = 1;
9   }
10 }
```

COBOL は 10 進数固定小数点形式で値を格納するため，例えば 0.1 を 0.1 として解釈して変数に格納する [日本規 11]．そのため，COBOL のプログラム (プログラム 1.1, 2 ページ) の 8 行目の条件分岐で必ず真となる．C 言語は 2 進数浮動小数点形式で値を格納する．0.1 を 2 進数浮動小数点形式で表現すると無限小数になるため格納時に丸めを行うが，C 言語では丸めの方向は処理系やアーキテクチャに依存する仕様のため，処理系やアーキテクチャを仮定する必要がある [日本規 03]．そのため，C 言語のプログラム (プログラム 1.2, 2 ページ) の 4 行目の条件分岐で真にならない場合がある．つまり，固定小数点数を浮動小数点数で代用した移植を行う時，真の値が得られなくなる点やそれに伴う条件分岐における条件式の真偽が実行環境に応じて変わる場合がある．

値の格納形式を例に挙げてソースコードの解析のみでは等価性の定式化は不完全であることを示したが，実際にはプログラミング言語や処理系，アーキテクチャの仕様の変化にしたがって動作が変化する場合がある．そこで，2 つのプログラムの等価性を，「全ての実行環境で全くおなじ計算パスを通り，値も完全に一致する」という意図で定めると，全ての実行環境を定式化しなければならない．また，「最後の計算結果だけある程度一致していればよい」という意図で定めても，複数の実行環境を想定して定式化する必要がある．固定小数点数を浮動小数点数で代用した移植ではその点のみを比べて等価性を求めたいが，アーキテクチャや処理系の仕様に基づいた実行環境の定式化も必要になる．つまり，それぞれのプログラミング言語について想定される実行環境に応じてソースコードに表われる定数記号や関数記号の解釈を与える必要がある．

等価性検証器の開発者からみると，プログラミング言語や処理系やアーキテクチャの全ての仕様の組み合わせを考慮しながら定式化することは膨大な時間と労力がかかり，実行環境の一要素が開発，発表されるたびに新たに等価性検証器を開発する必要がある．また，過去に作られたプログラムの実行環境に関する仕様書の紛失や公開停止によって定式化ができない場合もある．使用者からみると，実行環境に対応する等価性検証器を正しく選択することが求められ，等価性検証器が開発されていなければ等価性を求めることもできない．

1.2 研究目的

処理系やプログラミング言語，アーキテクチャの仕様の組み合わせを考慮しないプログラムの定式化とその定式化同士の等価性を定めることで検証器の開発者の負担が減り，変換前後のプログラム同士の等価性を議論することができる．そのため本研究では，区間論理とその上の区間遷移システムを提案し，遷移システムと区間遷移システムの間の関係を定式化する．区間論理は処理系やプログラミング言語，アーキテクチャの仕様の組み合わなど，複数の実行環境の定式化であり，区間遷移システムは複数の実行環境における 1 つのプログラムの実行の定式化である．更に，同一のプログラムによる遷移システムと区間遷移システムの間に成立する関係を証明する．

1.3 論文の構成

本論文の構成は次のとおりである．2 章では先行研究としてプログラムの定式化のための多ソート一

階述語論理とその上の遷移システム，計算の精度向上を図るための区間解析について説明する．３章では本研究で提案する遷移システムと区間論理，区間遷移システムの定義や区間遷移システムと遷移システムの間の関係を説明し，同一のプログラムから得られる遷移システムと区間遷移システムの間に成り立つ性質を証明する．４章で結論と課題を述べる．

第 2 章

関連研究

2.1 準備: 多ソート一階述語論理

2.1.1 プログラムと実行環境の定式化の例

型を持つプログラムをモデル化する方法の一つとして多ソート一階述語論理が挙げられる [Gal15]. これは，プログラムの実行環境を表現できる.

ここでは, 実行環境の例を 2 つ示す. 実行環境 1 では値を 0.5 区切りで格納し, 実行環境 2 では値を 0.1 区切りで格納することを仮定する. つまり, 定数記号 a を実行環境 1 では $\mathrm{round}_1(a) = \lfloor a \times 0.5^{-1} \rfloor \times 0.5$ として解釈し，実行環境 2 では $\mathrm{round}_2(a) = \lfloor a \times 0.1^{-1} \rfloor \times 0.1$ として解釈すると仮定する．また，関数記号 f を実行環境 1 では $\mathrm{round}_1(f(a,b))$ として解釈し，実行環境 2 では $\mathrm{round}_2(f(a,b))$ として解釈すると仮定する．このとき，C 言語のプログラム (プログラム 1.2, 2 ページ) 中にあらわれる定数記号は実行環境の例 (表 2.1, 5 ページ) となる.

表 2.1　実行環境の例

定数記号	実行環境 1 における記号の解釈	実行環境 2 における記号の解釈
a	$\mathrm{round}_1(a)$	$\mathrm{round}_2(a)$
`0.1`	0	0.1
`0.4`	0	0.4
`0.5`	0.5	0.5

2.1.2 各定義

まずは，多ソート一階述語論理の記号を定める.

定義 2.1.2.1 (言語)
多ソート一階述語論理の言語を $\mathcal{S} = (S, V, C)$ で表す.

- S は基本型の有限集合
- $T = S \cup \{t_1, \cdots, t_n; t \mid n \in \mathbb{N}, t_1, \cdots, t_n, t \in S\}$
- $V = \{V_t\}_{t \in S}$ は変数記号の集合
- $C = \{C_t\}_{t \in T}$ は定数記号の集合

次に，記号の集合を用いて式を定める．

定義 2.1.2.2 (式)
変数の集合 $X \subseteq V$ について，以下の条件を満たす型付きの表現の和集合 $E(X) = \{E_t(X)\}_{t \in T}$ のことを式と定める．

- 各 $t \in S$ について $v \in X, v \in V_t \Rightarrow v \in E_t(X)$
- 各 $t \in S$ について $c \in C_t \Rightarrow c \in E_t(X)$
- $c \in C_{t_1,\cdots,t_n;t}, e_1 \in E_{t_1}(X), \cdots, e_n \in E_{t_n}(X) \Rightarrow c(e_1, \cdots, e_n) \in E_t(X)$

定義 2.1.2.3 (主張)
変数の集合 $X \subseteq V$ について，以下の条件を満たす式 $A(X)$ のことを主張と定める．

- $E_{\mathrm{bool}}(X) \subseteq A(X)$
- $c \in C_{\mathrm{bool},\cdots,\mathrm{bool};\mathrm{bool}}, \varphi_1 \in A(X), \cdots, \varphi_n \in A(X) \Rightarrow c(\varphi_1, \cdots, \varphi_n) \in A(X)$
- $v \in X, \varphi \in A(X) \forall v : \varphi \in A(X)$
- $v \in X, \varphi \in A(X) \exists v : \varphi \in A(X)$

最後に解釈を定める．

定義 2.1.2.4 (解釈)
以下の条件を満たす組 $(\{D_t\}_{t \in S}, \{I_{t,c}\}_{t \in T, c \in C_t}, \{I_{t,v}\}_{t \in T, v \in V_t})$ のことを解釈と定める．

- $\{D_t\}_{t \in S}$ は対象集合
- $\{I_{t,c}\}_{t \in S, c \in C_t} \in C_t \to D_t$ は定数記号から対象への写像
- $\{I_{t,v}\}_{t \in S, v \in V_t} \in V_t \to D_t$ は変数記号から対象への写像

それぞれ以下のように拡張できる．

- $D_{t_1,\cdots,t_n;t} = D_{t_1} \times \cdots \times D_{t_n} \to D_t$
- $\forall t \in T, \forall c \in C_t \ I_c^t : C_t \to D_t$
- $\forall t \in T, \forall v \in V_t \ I_v^t : V_t \to D_t$

2.2 遷移システム

多ソート一階述語論理の上の遷移システムを構成する手法として，Pnueli らは変換器検証器の等価性証明のための遷移システムや反応システム及び並行システムのための遷移システムを定義している[PSS98][MP92]．状態を変数記号の解釈と定め，プログラムから生成した多ソート一階述語論理の主張を満足する状態の対の間に遷移があるような状態遷移である．状態遷移の中から無限列を 1 つとりだしたものを計算と定めている．状態はプログラムが取り得る実行状態，遷移はステップ間の関係，計算はプログラムの実行を表現できる．

例えば C 言語のプログラム (プログラム 1.2, 2 ページ) は初期状態を式. (2.1)，遷移関係を式. (2.2) と表わせる．

$$\Theta = (x = 0.1 \wedge y = 0.4 \wedge l = 3) \tag{2.1}$$

$$\rho = \begin{cases} (x + y < 0.5 \wedge l = 3 \wedge l' = 4) \\ \vee(\neg(x + y < 0.5) \wedge l = 3 \wedge l' = 7) \\ \vee(x' = (x + y)/0.5 \wedge l = 4 \wedge l' = 5) \\ \vee(x' = 1 \wedge l = 7 \wedge l' = 8) \\ \vee((l = 5 \vee l = 8) \wedge l = 9) \\ \vee(l = 9 \wedge l' = 9) \end{cases} \tag{2.2}$$

各実行環境の例 (表 2.1, 5 ページ) における計算を求めると実行環境 1 における計算 (図 2.1, 8 ページ) や実行環境 2 における計算 (図 2.2, 8 ページ) のように表わせる．遷移関係 ρ はこれら 2 つの計算の存在を保証しており，計算はプログラムの各状態における変数の実行環境での値と実行パスを表現する．

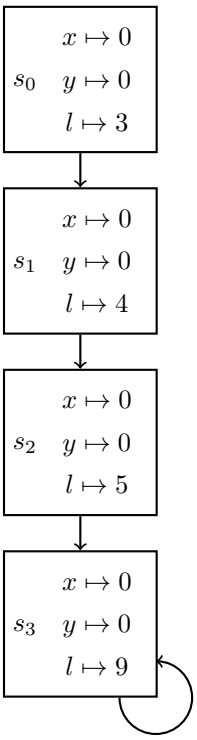

図 2.1　実行環境 1 における計算

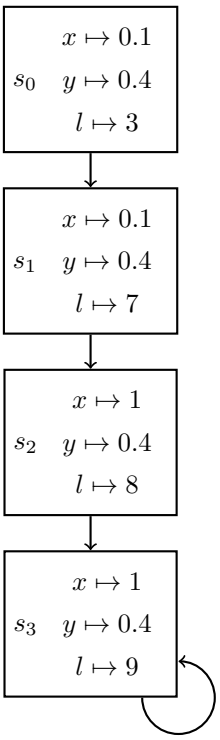

図 2.2　実行環境 2 における計算

実行環境 1 では 4 行目における条件式 `x + y < 0.5` の解釈は $0 + 0 < 0.5$ であり，結果真となるため 5 行目に遷移する．実行環境 2 では 4 行目における条件式 `x + y < 0.5` の解釈は $0.1 + 0.4 < 0.5$ であり，結果偽となるため 7 行目に遷移する．

これを [PSS98] では次のように定式化している．

定義 2.2.1 (遷移システム)
V, Θ, ρ, E が以下の条件を満たす時に $\mathcal{A} = (V, \Theta, \rho, E)$ を遷移システムと定める．

変数記号の集合 V
初期状態 $\Theta \in A(\Pi)$
遷移関係 $\rho \in A(\Pi \cup \Pi')$
プログラム中に表われる変数記号の集合 $E \subseteq V$

定義 2.2.2 (遷移システムの計算)
無限列 $\sigma = s_0, s_1, \ldots (s_i \in \Sigma)$ が $\mathcal{A} = (V, \Theta, \rho, E)$ の計算であるとは，以下の条件を満たすときである．

初期状態 $s_0 \vDash \Theta$
遷移関係 すべての $i \in N$ について $(s_i, s_{i+1}) \vDash \rho$

さらに，Pnueli らは遷移システムを用いてコンパイラやトランスパイラなどの変換器の検証方法を提案しており [PSS98]，遷移システムや変換器の検証方法を実装している例がいくつかある [SMK13][Nec00].

2.3 プログラム解析

データフロー解析はプログラムをグラフとして捉える解析方法で，各ノードはコードの一要素であり，各ノード間のネットワークはコードの一要素からどのように制御が渡されるかを記述する．各命令にラベルを与え，命令の実行前後で変数の取り得る値を集合に加えてプログラム全体における変数の値の流れを表現できる．コンパイラの最適化に用いられる [NNH05].

2.4 区間解析

計算機では真の値を用いて計算できない仕様が存在し，様々な言語でその仕様が採用される．そこで，計算機で表現可能な値を最小値と最大値とし，真の値を持つ区間とその上の演算を定めることで真の計算結果が必ず区間上の計算結果の中に存在するような代数を生成できる．これを用いて計算の精度を求めることもできる [Nic07].

2.5 提案手法との違い

関連研究では各プログラミング言語，処理系，アーキテクチャなどの仕様の組み合わせによって遷移システムの基礎となる解釈を定義し直す必要がある．検証器の開発者は関連研究を用いた定式化の実装時にすべての仕様を確認すべきであり，実装されたツールを使うユーザはその仕様の組み合わせに当てはまるものかを確認する必要がある．すべての仕様の組は膨大であるために実装コストがかかることや文書が残されていないものに関して実装不可能である．

また，区間解析は関数に関する精度を保証するがプログラムの各変数に対する精度の保証の為には関数に変換する必要がある．

第 3 章

提案手法

3.1　提案手法の概要と本章の構成

本研究では，各型の各記号の解釈が区間であり，その区間の最小値と最大値が計算機が表現可能な対象集合の要素であるものを採用し，プログラム中の論理と計算を保証する論理を分けた区間論理とプログラムの表現である区間遷移システムを提案する．区間を採用したことで，プログラム中の論理と計算を保証する論理プログラミング言語や様々な処理系，アーキテクチャなど，様々な仕様を満たす値を含む遷移システムを集約できる．

第 3.2 節では具体例を記しながら提案手法の第 3.3 節以降の定義における工夫点を記述する．そして第 3.3 節以降では定義や定理を羅列する．複数の実行環境を表現するために遷移システムと区間遷移システムの構文で満たすべき制約を第 3.3 節で示す．第 3.3 節で制限した構文に対する 1 つの実行環境や複数の実行環境を定式化した解釈を第 3.4 節で与える．最後に区間遷移システムが遷移システムを模倣することを第 3.5 節で示す．

3.2　提案手法の工夫点と例

研究目的を達成する構成方法を説明するために，まずはプログラムの実行の系列を論理式と状態遷移で表現する素朴な集約の例を示す．そして，素朴な集約に問題があることを示し，提案手法の例を与えながら定義や定理の位置を示す．

3.2.1　提案手法と異なる素朴な集約

研究目的である複数の実行環境の定式化を達成するための 1 つの構成方法として，遷移システムのみを用いてこれらを集約して表現することを考える．すると状態の要素である各変数記号の解釈が全ての遷移システムの各状態における各変数記号の解釈として取り得る値を含む閉空間と定める方法が挙げられる．具体的に表 2.1 を用いて構成すると，図 3.1 となるような定義を考える．

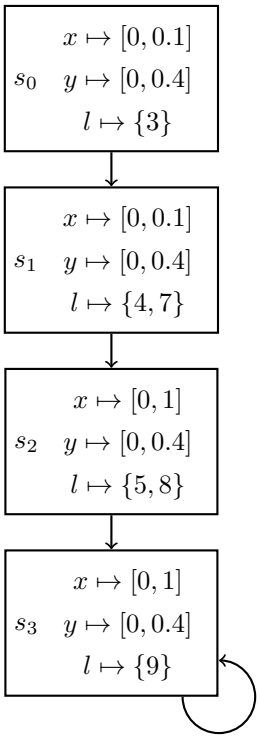

図 3.1　本研究の提案手法と異なる方法で集約された計算の構成例

しかし，これには 2 つ問題点がある．

1 つめは，解釈として採用した閉空間の最小値と最大値を，等価性を証明するための実行環境で表現できない場合がある．例えば COBOL のプログラムの小数点数を C 言語の double 型を用いて表現した場合が挙げられる．

次に，計算が列になるとは限らず計算の定義定義. 2.2.1 に反して扱いづらい場合がある．図 3.1 の例では定数記号 a の解釈を $[\min\{\mathrm{round}_1(a), \mathrm{round}_2(a)\}, \max\{\mathrm{round}_1(a), \mathrm{round}_2(a)\}]$ とする．このとき x の解釈は $[0, 0.1]$，y の解釈は $[0, 0.4]$，0.5 の解釈は $[0.5]$ となり，関数 $+$ を適用した結果の解釈は $[0, 0.5]$ となる．分岐後の命令の個数が一致しているため構成できているが，そうでない場合，状態の列が分岐したり新たな定式化が必要になる．

3.2.2　提案手法の工夫点

以上の問題を解決するために以下の条件を満たす定式化が必要がある．

(1) 各変数記号の解釈は証明するための実行環境で表現できる
(2) 集約したどの計算もひとつの論理式を満足する

そこで，本研究ではプログラムを表現する構文 (定義. 3.3.1, 16 ページ) に工夫を 1 つ，1 つの実行環境を表現する論理 (定義. 3.4.2.1, 17 ページ) と複数の実行環境を模倣できる区間論理 (定義. 3.4.3.1, 18 ページ) に共通して工夫を 1 つ加えた．構文 (定義. 3.3.1, 16 ページ) に与えた工夫は，プログラム中の条件式を表現する構文とプログラムを表現する構文をソートから分けたことであり，論理 (定義. 3.4.2.1, 17 ページ) と区間論理 (定義. 3.4.3.1, 18 ページ) に与えた工夫は Com の解釈を具体的に定義した点と，区間論理 (定義. 3.4.3.1, 18 ページ) における対象集合を，[Nic07] を参考に，証明する実

行環境で表現可能な値が最小値と最大値となる閉空間の集合とした点である．

まずは (1) に関しては区間論理 (定義. 3.4.3.1, 18 ページ) における対象集合を，[Nic07] を参考に，表現可能な集合 F を定めて想定した全ての実行環境の下界上界のうち，F の要素で最小値と最大値を取ってそれを閉区間の最小値および最大値となるよう構成することで表現できる．証明する実行環境で表現可能な値が最小値と最大値となる閉空間の集合としたことで (1) の条件を満たしている．

(2) に関連する工夫を記述する前に，素朴な定式化における問題の原因を記述する．条件式 $x+y<0.5$ は実行環境 1 における計算 (図 2.1, 8 ページ) は真で別の実行環境 2 における計算 (図 2.2, 8 ページ) では偽であった．式. (2.1) や式. (2.2) はどちらの実行環境でも真であった．つまり，プログラム中の条件式の解釈は実行環境が変われば真偽も変わるが，プログラムを表現する論理式はある実行環境において計算が存在すれば真であるという違いがある．

よって，本研究では性質の異なる 2 つの論理を構文から分けて圧縮する工夫を定義に与えた．構文 (定義. 3.3.1, 16 ページ) でプログラム中の論理のためのソートを boolean，プログラムを表現する論理のためのソートを bool と定義している．また，プログラム中の論理をプログラムを表現する論理に合わせるための関数記号 $\mathrm{Com} : C_{\mathrm{boolean}} \to C_{\mathrm{bool}}$ を定義する．そして，プログラム中の論理のための記号 C_{boolean} 及び Com に対する解釈を論理 (定義. 3.4.2.1, 17 ページ) と区間論理 (定義. 3.4.3.1, 18 ページ) で切り変わるように定める．boolean の対象領域や各論理における解釈 (表 3.1, 13 ページ) は論理 (定義. 3.4.2.1, 17 ページ) や区間論理 (定義. 3.4.3.1, 18 ページ) における C_{boolean} 及び Com に対する解釈を具体的に示したものである．そうして定めた論理と区間論理の間の関係を区間表現 (定義. 3.4.3.2, 19 ページ) と区間表現可能 (定義. 3.4.3.3, 19 ページ) のように定めて区間表現可能な区間論理の構成 (定理. 3.4.3.2, 20 ページ) を証明することで実行環境の集約は仕様が参照できない実行環境も集約できることを保証する．最後に論理の上に [PSS98] や [MP92] を参考にして遷移システム (定義. 3.5.1.2, 21 ページ) を定め，論理の上の遷移システムと区間表現可能な区間論理の構成 (定理. 3.4.3.2, 20 ページ) の上の遷移システムの間に区間表現模倣性 (定義. 3.5.2.1, 21 ページ) を定め，具体的な構成である区間表現可能な区間論理の構成 (定理. 3.4.3.2, 20 ページ) の上の遷移システムの計算が定式化できない実行環境の遷移システムの計算を模倣することを区間表現可能模倣性 (定理. 3.5.2.1, 22 ページ) として示す．

表 3.1　boolean の対象領域や各論理における解釈

名称	記号	論理	区間論理
対象領域	D_{boolean}	$\{\bigcirc, \times\}$	$\mathcal{P}(\{\bigcirc, \times\}) \setminus \emptyset = \{[\times], [\bigcirc], [\times, \bigcirc]\}$
表現可能集合	F_{boolean}	不要	$\{\bigcirc, \times\}$
順序	$\leq_{\mathrm{boolean}}$	$\{(\times, \times), (\times, \bigcirc), (\bigcirc, \bigcirc)\}$	不要
真	true	$\bigcirc$	$[\bigcirc]$
偽	false	$\times$	$[\times]$
否定	!	$\begin{pmatrix} \bigcirc \mapsto \times \\ \times \mapsto \bigcirc \end{pmatrix}$	$\begin{pmatrix} [\times, \bigcirc] \mapsto [\times, \bigcirc] \\ [\bigcirc] \mapsto [\times] \\ [\times] \mapsto [\bigcirc] \end{pmatrix}$
論理和	|	$\begin{pmatrix} (\bigcirc, \bigcirc) \mapsto \bigcirc \\ (\bigcirc, \times) \mapsto \bigcirc \\ (\times, \bigcirc) \mapsto \bigcirc \\ (\times, \times) \mapsto \times \end{pmatrix}$	| | $[\times]$ | $[\bigcirc]$ | $[\times, \bigcirc]$ | | $[\times]$ | $[\times]$ | $[\bigcirc]$ | $[\times, \bigcirc]$ | | $[\bigcirc]$ | $[\bigcirc]$ | $[\bigcirc]$ | $[\times, \bigcirc]$ | | $[\times, \bigcirc]$ | $[\times, \bigcirc]$ | $[\times, \bigcirc]$ | $[\times, \bigcirc]$ |
論理積	&	$\begin{pmatrix} (\bigcirc, \bigcirc) \mapsto \bigcirc \\ (\bigcirc, \times) \mapsto \times \\ (\times, \bigcirc) \mapsto \times \\ (\times, \times) \mapsto \times \end{pmatrix}$	| | $[\times]$ | $[\bigcirc]$ | $[\times, \bigcirc]$ | | $[\times]$ | $[\times]$ | $[\times]$ | $[\times]$ | | $[\bigcirc]$ | $[\times]$ | $[\bigcirc]$ | $[\times, \bigcirc]$ | | $[\times, \bigcirc]$ | $[\times]$ | $[\times, \bigcirc]$ | $[\times, \bigcirc]$ |
圧縮	Com	$\begin{pmatrix} \bigcirc \mapsto \top \\ \times \mapsto \bot \end{pmatrix}$	$\begin{pmatrix} [\times, \bigcirc] \mapsto \top \\ [\bigcirc] \mapsto \top \\ [\times] \mapsto \bot \end{pmatrix}$

3.2.3　提案手法を用いた具体例と提案手法が与える利点

論理によって Com の解釈を切り変えることでうける利点をプログラム 1.2 を例に挙げて記述する．プログラムを表現する論理式は，初期状態は式. (2.1) と変わらず，遷移関係は式. (3.1) となる．

$$\rho = \begin{cases} (\mathrm{Com}(x + y < 0.5) \wedge l = 3 \wedge l' = 4) \\ \vee(\mathrm{Com}(!(x + y < 0.5)) \wedge l = 3 \wedge l' = 7) \\ \vee(x' = (x + y)/0.5 \wedge l = 4 \wedge l' = 5) \\ \vee(x' = 1 \wedge l = 7 \wedge l' = 8) \\ \vee((l = 4 \vee l = 8) \wedge l = 9) \end{cases} \tag{3.1}$$

1 つの実行環境を表現する論理 (定義. 3.4.2.1, 17 ページ) ではプログラムの条件式の解釈は一意に定まり，プログラムを表現する論理式の真偽と一致するため，Com を用いて，プログラムの条件式の解釈が真ならばプログラムを表現する論理式の解釈も真をとり，プログラムの条件式の解釈が偽ならばプログラムを表現する論理式の解釈も偽をとる．[PSS98] や [MP92] ではこの性質からこの 2 つを同一視して定式化している．一方で，複数の実行環境を表現する区間論理 (定義. 3.4.3.1, 18 ページ) ではプログラムの条件式の解釈にも区間を採用し，Com を用いて，プログラムの条件式の解釈がどんな実行環境でも真ならばプログラムを表現する論理式の解釈も真を返し，プログラムの条件式の解釈がどんな実行環境でも偽ならばプログラムを表現する論理式の解釈も偽を返し，プログラムの条件式の解釈が実行

環境に依存して結果がかわるならばプログラムを表現する論理式の解釈を真におしつぶして返すことで一方の条件を満たす計算も他方の条件を満たす計算も 1 つの論理式で保証できる．

加えて，2 つの工夫によって実行環境の仕様が参照できないとき実行環境を定式化できない問題も解決できる．仕様が参照できない実行環境における記号の解釈が集約された実行環境の記号の解釈の要素であることがいえれば実行環境の集約は仕様が参照できない実行環境も集約でき，その条件の上で仕様が参照できない実行環境の計算はある実行環境の集約の上の計算が模倣している．

以上を考慮して C 言語のプログラム (プログラム 1.2, 2 ページ) について，計算の存在を保証する各論理式を再構成し，実行する 2 つの実行環境表 2.1 を集約した区間論理とその上の区間遷移システムの具体例を以下に 2 つ示す．まずは複数の実行環境を集約する，区間論理 $I\mathcal{L}_1$ と $I\mathcal{L}_2$ を定義する．区間論理 $I\mathcal{L}_1$ における表現可能集合を $F = \{a \in D_{\mathrm{double}} \mid \forall x \in D_{\mathrm{double}}(a = x_1)\}$，区間論理 $I\mathcal{L}_2$ における表現可能集合を $F = \{a \in D_{\mathrm{double}} \mid \forall x \in D_{\mathrm{double}}(a = x_2)\}$ と定めると C 言語のプログラム (プログラム 1.2, 2 ページ) に表われる定数記号は表 3.2 のように解釈できる．

表 3.2　実行環境の例

記号	区間論理 $I\mathcal{L}_1$ における記号の解釈	区間論理 $I\mathcal{L}_2$ における記号の解釈
0.1	$[0, 0.5]$	$[0, 0.1]$
0.4	$[0, 0.5]$	$[0, 0.4]$
0.5	$[0.5]$	$[0.5]$

論理 (定義. 3.4.2.1, 17 ページ) によって与えた解釈を `x + y < 0.5` に用いると，実行環境 1 の論理では ◯，実行環境 2 の論理では × であり，両方の実行環境を集約した区間論理では $[\times, \bigcirc]$ となる．そして，$\mathrm{Com}(x + y < 0.5)$ の解釈は実行環境 1 の論理では $\top$，実行環境 2 の論理では $\bot$ であり，両方の実行環境を集約した区間論理では $\top$ となる．また，$\mathrm{Com}(!(x + y < 0.5))$ の解釈は実行環境 1 の論理では $\bot$，実行環境 2 の論理では $\top$ であり，両方の実行環境を集約した区間論理では $\top$ となる．すると，図 2.1 も図 3.2 や図 3.4 のような計算も共通の式. (2.1) と式. (3.1) によって保証され，どちらの計算も，各状態のすべての変数記号の解釈も，図 2.1 の対応する状態における同じ変数記号を含んでいる．同様に，図 2.2 も図 3.3 や図 3.5 のような計算も共通の式. (2.1) と式. (3.1) によって保証され，どちらの計算も，各状態のすべての変数記号の解釈も，図 2.2 の対応する状態における同じ変数記号を含んでいる．つまり，提案手法を用いると 1 つのプログラムの定式化があれば複数の実行環境の定式化におけるプログラムの実行を表現できることがわかり，研究目的を満たす．

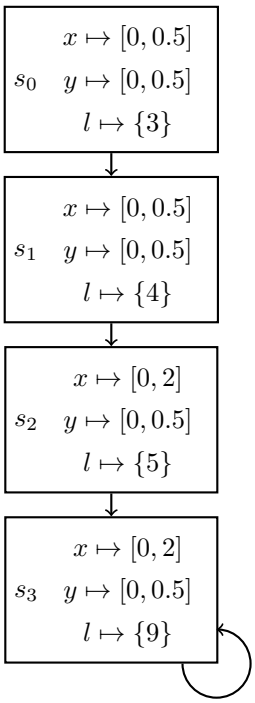

図 3.2　区間論理 $I\mathcal{L}_1$ における真を通るパスを表現する計算

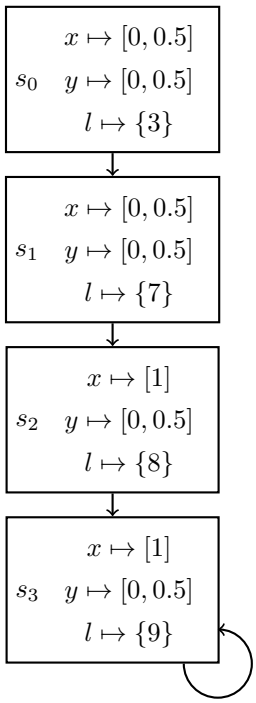

図 3.3　区間論理 $I\mathcal{L}_1$ における偽を通るパスを表現する計算

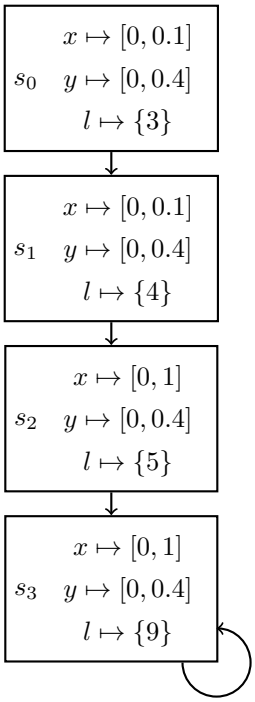

図 3.4　区間論理 $I\mathcal{L}_2$ における真を通るパスを表現する計算

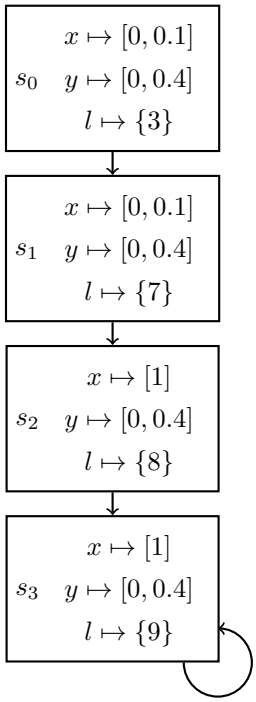

図 3.5　区間論理 $I\mathcal{L}_2$ における偽を通るパスを表現する計算

3.3 構文

遷移システムと区間遷移システムの間で共通する構文に制約を与える．boolean はプログラム中に現れる論理の型で，否定と積和の記号を導入することでプログラム中の論理を表現する．bool は計算を表現する主張に現れる論理の型である．boolean と bool を分けることで，区間論理で複数の実行環境を表現できる．また，boolean から bool へ圧縮する関数 Com を定めて論理と区間論理に必ず含まれるように制限することで 1 つのプログラムの実行を表現できる．また，location を導入することで条件分岐やジャンプを表現できる．

定義 3.3.1 (構文)
構文 $\mathcal{S} = (S, C)$ は以下の性質を持つ．

location の型 $\mathrm{location} \in S$
プログラム中に現れる論理の型 $\mathrm{boolean} \in S$
プログラム中に現れる真偽 $C_{\mathrm{boolean}} = \{\mathrm{true}, \mathrm{false}\}$
プログラム中に現れる否定 $! \in C_{\mathrm{boolean:boolean}}$
プログラム中に現れる積和 $\{\&, |\} \subseteq C_{\mathrm{boolean,boolean:boolean}}$
状態の列を表現する論理の型 $\mathrm{bool} \in S$
状態の列を表現する論理に現れる真偽 $C_{\mathrm{bool}} = \{\top, \bot\}$
状態の列を表現する論理に現れる否定 $\neg \in C_{\mathrm{bool:bool}}$
状態の列を表現する論理に現れる積和 $\{\wedge, \vee\} \subseteq C_{\mathrm{bool,bool:bool}}$
等号 $\forall t \in S \ :=_{t,t:\mathrm{bool}} \in C_{t,t:\mathrm{bool}}$

3.4 論理と区間論理

構文 (定義. 3.3.1, 16 ページ) に解釈を与える. bool はプログラムの定式化のためのソートで, boolean はプログラム中に現れる論理のためのソートである.

3.4.1 共通する解釈

1 つの実行環境におけるプログラムの定式化のためのソート bool に関する解釈はどの実行環境においても一意だから 1 つに定める.

定義 3.4.1.1 (bool の解釈)

真偽 $D_{\mathrm{bool}} = \{\top, \bot\}$

否定 $[\![\neg]\!] = \begin{pmatrix} \top \mapsto \bot \\ \bot \mapsto \top \end{pmatrix}$

和 $[\![\vee]\!] = \begin{pmatrix} (\top,\top) \mapsto \top \\ (\top,\bot) \mapsto \top \\ (\bot,\top) \mapsto \top \\ (\bot,\bot) \mapsto \bot \end{pmatrix}$

積 $[\![\wedge]\!] = \begin{pmatrix} (\top,\top) \mapsto \top \\ (\top,\bot) \mapsto \bot \\ (\bot,\top) \mapsto \bot \\ (\bot,\bot) \mapsto \bot \end{pmatrix}$

3.4.2 論理

構文 (定義. 3.3.1, 16 ページ) の各記号に対して表 3.1 を満たすように意味を与える. 1 つの実行環境を考えた時, プログラム中の論理 boolean は実行環境ごとに一意だから bool と boolean の結果を一致させるように Com は ◯ なら $\top$, × なら $\bot$ に写すように定める.

定義 3.4.2.1 (論理)

$\mathcal{S}_{\mathcal{L}}$, $\mathcal{I}_{\mathcal{L}}$ が以下の条件を満たすとき, $(\mathcal{S}_{\mathcal{L}}, \mathcal{I}_{\mathcal{L}})$ を論理 $\mathcal{L}$ と定める.

- 構文 $\mathcal{S}_{\mathcal{L}}$ は構文 (定義. 3.3.1, 16 ページ) に従う
- 解釈 $\mathcal{I}_{\mathcal{L}} = (\{D_t\}_{t\in S}, \{I_t\}_{t\in T})$ は以下の性質を持つ
  対象領域 すべての $t \in S \setminus \{\text{location, bool}\}$ の D_t について全順序 $\leq_t$ が定義されている.
  boolean **の対象領域** $D_{\mathrm{boolean}} = \{◯, ×\}$
  boolean **の順序** $\leq_{D_{\mathrm{boolean}}} = \{(×,×), (×,◯), (◯,◯)\}$
  boolean **の真** $[\![\mathrm{true}]\!] = ◯$
  boolean **の偽** $[\![\mathrm{false}]\!] = ×$
  boolean **の否定** $[\![!]\!] = \begin{pmatrix} ◯ \mapsto × \\ × \mapsto ◯ \end{pmatrix}$

boolean **の和** $[\![|]\!] = \begin{pmatrix} (\bigcirc,\bigcirc) \mapsto \bigcirc \\ (\bigcirc,\times) \mapsto \bigcirc \\ (\times,\bigcirc) \mapsto \bigcirc \\ (\times,\times) \mapsto \times \end{pmatrix}$

boolean **の積** $[\![\&]\!] = \begin{pmatrix} (\bigcirc,\bigcirc) \mapsto \bigcirc \\ (\bigcirc,\times) \mapsto \times \\ (\times,\bigcirc) \mapsto \times \\ (\times,\times) \mapsto \times \end{pmatrix}$

boolean **から** bool **への圧縮** $[\![\mathrm{Com}]\!] = \begin{pmatrix} \bigcirc \mapsto \top \\ \times \mapsto \bot \end{pmatrix}$

location **の対象集合** $D_{\mathrm{location}} = \mathbb{N}$

bool bool $\in S$ について bool の解釈 (定義. 3.4.1.1, 17 ページ) をみたす

3.4.3 区間論理

区間論理でも同様に構文 (定義. 3.3.1, 16 ページ) の解釈を与える．区間論理では，bool と location を除く，boolean を含めた全ての記号の解釈を区間として与え，bool は bool の解釈 (定義. 3.4.1.1, 17 ページ) で定めた解釈を用いる．プログラムの中の条件分岐で boolean から bool へ圧縮する関数 Com の解釈を，表 3.1 のように真を含む区間ならば真を返し，それ以外ならば偽を返すように定める．

この定義はある実行環境におけるある計算のパスが存在する時に真を返し，どの実行環境でもある計算パスが存在しない時に偽を返すように設計している．また，プログラム中に現れる論理記号の解釈は $t_0 =$ boolean における関数記号の解釈となる．抽象的に定義することで，例えばプログラム中の論理和を誤って論理積として実装された実行環境も考慮できる．

定義 3.4.3.1 (区間論理)

以下の条件を満たす論理 $I\mathcal{L} = (\mathcal{S}_{I\mathcal{L}}, \mathcal{I}_{I\mathcal{L}})$ を区間論理と定める．

- 構文 $\mathcal{S}_{I\mathcal{L}}$ は構文 (定義. 3.3.1, 16 ページ) に従う
- 解釈 $\mathcal{I}_{\mathcal{L}} = (\{D_t\}_{t\in S}, \{F_t\}_{t\in S}, \{I_t\}_{t\in T})$ は以下の性質を持つ

  台集合 すべての $t \in S \setminus \{\text{location, bool}\}$ における D_t について全順序 $\leq_t$ が定義されている．

  表現可能集合 $\forall t \in S\ F_t \subseteq D_t$

  対象領域 $(\forall t \in S\ ID_{t_{I\mathcal{L}}} = \{[a,b] \subseteq D_t | a \in F_t \wedge b \in F_t\}$

  定数記号 $(\forall t \in S \forall c_t \in C_t\ [\![c_t]\!]_{\mathcal{I}_{I\mathcal{L}}}) \in ID_{t_{I\mathcal{L}}}$

  関数記号 $\forall t \in S \forall i \leq n \forall t_i \in S \forall f \in C_{t_1,\ldots,t_n;t_0}\ [\![f]\!]_{I\mathcal{L}} :: \Pi_{1\leq i\leq n} C_{t_i} \to C_{t_0}$

  boolean **から** bool **への圧縮** $[\![\mathrm{Com}]\!] = \begin{pmatrix} [\times] \mapsto \bot \\ [\bigcirc] \mapsto \top \\ [\times,\bigcirc] \mapsto \top \end{pmatrix}$

  location **の対象集合** $ID_{\mathrm{location}} = \{\{a\} \in \mathcal{P}(\mathbb{N})\}$

  location **の解釈** $\forall l \in C_{\mathrm{location}}\ ([\![l]\!]_{I\mathcal{L}} = \{l\})$

  等号 $\forall t \in S, \forall x_t, x'_t \in ID_t\ [\![=]\!]_{II_{t,t:\mathrm{bool}}}(x_t, x'_t) = (x_t = x'_t)$

  bool bool $\in S$ について bool の解釈 (定義. 3.4.1.1, 17 ページ) をみたす

論理 $\mathcal{L}$ と区間論理 $I\mathcal{L}$ の間の関係を定める．実行環境の集約の定式化である区間論理 $I\mathcal{L}$ が 1 つの実行環境の定式化である論理 $\mathcal{L}$ をうまく集約していることを表す．

定義 3.4.3.2 (区間表現)
論理 $\mathcal{L}$ と区間論理 $I\mathcal{L}$ の間で以下の式が成り立つとき，区間論理 $I\mathcal{L}$ は論理 $\mathcal{L}$ の区間表現であると定める．

- $\mathcal{S}_{\mathcal{IL}} = \mathcal{S}_{\mathcal{L}}$
- $\forall t \in S \forall c_t \in C_t \ [\![c_t]\!]_{\mathcal{L}} \in [\![c_t]\!]_{I\mathcal{L}}$
- $\forall n \in \mathbb{N} \forall i \leq n \forall t_i \in S \forall f \in C_{t_1,\ldots,t_n;t_0} \forall X_i \in C_{t_i} \ [\![f(X_1,\ldots,X_n)]\!]_{\mathcal{L}} \in [\![f(X_1,\ldots,X_n)]\!]_{I\mathcal{L}}$

論理集合 L と区間論理 $I\mathcal{L}$ の間の関係を定義する．これは実行環境の集約が複数の実行環境を模倣する実行環境であることを表す．

定義 3.4.3.3 (区間表現可能)
論理集合 L の全ての要素 $\mathcal{L}$ について区間表現である区間論理 $I\mathcal{L}$ を L について区間論理可能であると定める．

特別な場合として一点集合を定数記号の解釈とするものを定める．これはひとつの環境の区間論理による定式化である．なお，関数記号の解釈には $t_0 = \text{boolean}$ も含まれており，この定義は表 3.1 を満たす．

定義 3.4.3.4 (一点区間論理)
論理 $\mathcal{L}$ と区間論理 $I\mathcal{L}$ について，以下の条件を満すとき，$I\mathcal{L}$ は $\mathcal{L}$ の一点区間論理であると定める．

- $\mathcal{S}_{I\mathcal{L}} = \mathcal{S}_{\mathcal{L}}$
- $\forall t \in S_{I\mathcal{L}} \ D_{t_{\mathcal{L}}} = D_{t_{I\mathcal{L}}}$
- $\forall t \in S_{I\mathcal{L}} \ F_t$ は有限
- $\forall t \in S_{I\mathcal{L}} \forall c_t \in C_t \ [\![c_t]\!]_{I\mathcal{L}} = \{[\![c_t]\!]_{\mathcal{L}}\}$
- $\forall n \in \mathbb{N} \forall i \leq n \forall t_i \in S \forall f \in C_{t_1,\ldots,t_n;t_0} \forall X_i \in D_{t_i}$
  $[\![f]\!]_{I\mathcal{L}}(X_1,\ldots X_n)$
  $= [\max\{a \in F \mid \forall j \leq n \ x_j \in X_j a \leq [\![f]\!]_{\mathcal{L}}(x_1,\ldots x_n)\}$
  $, \min\{a \in F \mid \forall j \leq n \ x_j \in X_j [\![f]\!]_{\mathcal{L}}(x_1,\ldots x_n) \leq a\}]$

一点区間論理が区間表現であることを示す．

定理 3.4.3.1 (一点区間論理が区間表現である)
論理 $\mathcal{L}$ と区間論理 $I\mathcal{L}$ について，$I\mathcal{L}$ が $\mathcal{L}$ の一点区間論理であるとき，$I\mathcal{L}$ は $\mathcal{L}$ の区間表現である．

証明.
$I\mathcal{L}$ が $\mathcal{L}$ の一点区間論理 (定義. 3.4.3.4, 19 ページ) であることを仮定すると，$\mathcal{S}_{\mathcal{L}} = \mathcal{S}_{I\mathcal{L}}$，$\forall t \in S \forall c_t \in C_t \ [\![c_t]\!]_{\mathcal{L}} \in [\![c_t]\!]_{I\mathcal{L}}$ を満たす．ここで，$\forall n \in \mathbb{N} \forall i \leq n \forall t_i \in S \forall X_i \in C_{t_i} \forall f \in C_{t_1,\ldots,t_n;t_0}$ を満たす n, i, t_i, X_t, f を仮定すると，$\max\{a \in F \mid \forall j \leq n \forall x_j \in [\![X_j]\!]_{I\mathcal{L}} \ a \leq [\![f]\!]_{\mathcal{L}}(x_1,\ldots x_n)\} \leq [\![f(X_1,\ldots,X_n)]\!]_{\mathcal{L}}$ かつ $[\![f(X_1,\ldots,X_n)]\!]_{\mathcal{L}} \leq \min\{a \in F \mid \forall j \leq n \forall x_j \in [\![X_j]\!]_{I\mathcal{L}} \ [\![f]\!]_{\mathcal{L}}(x_1,\ldots x_n) \leq a\}$ だから $[\![f(X_0,\ldots,X_n)]\!]_{\mathcal{L}} \in [\![f(X_0,\ldots,X_n)]\!]_{I\mathcal{L}}$．よって $I\mathcal{L}$ は、$\mathcal{L}$ の区間表現である．

具体的に区間論理の 1 つの解釈を与え，区間論理が論理集合を区間表現可能であることを示す．この定理は，以下に示す区間論理が複数の実行環境を表現できることを意味する．また，関数記号の解釈は実装環境によって異なるため，すべての実行環境の関数をすべての実行環境で使われる引数を用いて得

られた結果を返すように実装している．なお，関数記号の解釈には $t_0 = \mathrm{boolean}$ も含まれており，この構成は表 3.1 を満たす．

定理 3.4.3.2 (区間表現可能な区間論理の構成)
論理集合 L と区間論理 $I\mathcal{L}$ について，以下の性質を満たす時，区間論理 $I\mathcal{L}$ は L について区間表現可能である．

- $\forall \mathcal{L} \in L\ \mathcal{S}_{I\mathcal{L}} = \mathcal{S}_{\mathcal{L}}$
- $\forall \mathcal{L} \in L \forall t \in S_{I\mathcal{L}}\ D_{t_{I\mathcal{L}}} = D_{t_{\mathcal{L}}}$
- $\forall t \in S_{I\mathcal{L}}\ F_t$ は有限
- $\forall t \in S_{I\mathcal{L}} \forall c_t \in C_t$
  $[\![c_t]\!]_{I\mathcal{L}} = [\max\{a \in \mathbb{F}_{I\mathcal{L}} \mid \forall \mathcal{L} \in La \leq [\![c_t]\!]_{\mathcal{L}}\}, \min\{a \in \mathbb{F}_{I\mathcal{L}} \forall \mathcal{L} \in L \mid [\![c_t]\!]_{\mathcal{L}} \leq a\}]$
- $\forall t \in S_{I\mathcal{L}} \forall i \leq n \forall t_i \in S_{I\mathcal{L}} \forall f \in C_{t_1,\ldots,t_n;t_0} \forall X_i \in ID_{t_{i\,I\mathcal{L}}}$
  $[\![f]\!]_{I\mathcal{L}}(X_{t_1}, \ldots X_{t_n})$
  $= [\max\{a \in \mathbb{F}_{I\mathcal{L}} \mid \forall \mathcal{L} \in L \forall j \leq n \forall x_j \in X_j\ a \leq [\![f]\!]_{\mathcal{L}}(x_1, \ldots, x_n)\},$
  $\min\{a \in \mathbb{F}_{I\mathcal{L}} \mid \forall \mathcal{L} \in L \forall j \leq n \forall x_j \in X_j\ [\![f]\!]_{\mathcal{L}}(x_1, \ldots, x_n) \leq a\}]$

証明.
論理集合 L と区間論理 $I\mathcal{L}$ について区間表現可能な区間論理の構成 (定理. 3.4.3.2, 20 ページ) を仮定するとすべての $\mathcal{L} \in L$ について以下の条件を満たす．

- $\mathcal{S}_{I\mathcal{L}} = \mathcal{S}_{\mathcal{L}}$
- $\forall t \in S \forall c_t \in C_t\ [\![c_t]\!]_{\mathcal{L}} \in [\![c_t]\!]_{I\mathcal{L}}$

ここで，$\forall n \in \mathbb{N} \forall i \leq n \forall t_i \in S_{I\mathcal{L}} \forall f \in C_{t_1,\ldots,t_n;t_0} \forall X_i \in ID_{t_{i\,I\mathcal{L}}}$ を満たす n, i, t_i, f, X_i をとる．$\max\{a \in \mathbb{F}_{I\mathcal{L}} \mid \forall \mathcal{L}' \in L \forall j \leq n \forall x_j \in X_j\ a \leq [\![f]\!]_{\mathcal{L}'}(x_1, \ldots, x_n)\} \leq [\![f(X_1, \ldots, X_n)]\!]_{\mathcal{L}}$ かつ $[\![f(X_1, \ldots, X_n)]\!]_{\mathcal{L}} \leq \min\{a \in \mathbb{F}_{I\mathcal{L}} \mid \forall \mathcal{L}' \in L \forall j \leq n \forall x_j \in X_j\ [\![f]\!]_{\mathcal{L}}(x_1, \ldots, x_n) \leq a\}$ だから $[\![f(X_0, \ldots, X_n)]\!]_{\mathcal{L}} \in [\![f(X_0, \ldots, X_n)]\!]_{I\mathcal{L}}$ よって $I\mathcal{L}$ は，すべての $\mathcal{L}$ の区間表現である．

3.5 遷移システムと区間遷移システム

3.5.1 遷移システム

遷移システムについて，論理関係と 2 つの状態を使って 1 ステップの前後の変数の状態の変化を表現できる．

定義 3.5.1.1 (状態の結合)
論理 $\mathcal{L}$ における変数記号の解釈 $s_{\mathcal{L}}, s'_{\mathcal{L}}$ を用いて $\rho \in A(\Pi \cup \Pi')$ を解釈する関数 $\triangleright$ を以下のように定める．

$$[\![\rho]\!]_{(s_{\mathcal{L}} \triangleright s'_{\mathcal{L}})} = \begin{cases} [\![\rho]\!]_{s_{\mathcal{L}}} & \rho \in \Pi \\ [\![\rho]\!]_{s'_{\mathcal{L}}} & \rho \in \Pi' \\ [\![\rho]\!]_{\mathcal{L}} & o.w. \end{cases}$$

遷移システムについて，各状態に遷移を与えるために論理式を解釈する方法を与える．プログラムの 1 つの実行全体を表現できる．また，元となる論理が区間論理であるとき，その区間遷移システムはプログラムの複数の実行環境を表現できる．

定義 3.5.1.2 (遷移システム)
論理 $\mathcal{L} = (\mathcal{S}, \mathcal{I})$ について，以下の条件を満たすような組 $\mathcal{A} = (\Pi_{\mathcal{L}}, \Sigma_{\mathcal{L}}, \rho_{\mathcal{L}}, \Theta_{\mathcal{L}})$ のことを，論理 $\mathcal{L}$ 上の遷移システムと定める．

変数記号 変数記号の集合を $\Pi \subseteq V$ と定める．
状態集合 状態の集合を $\Sigma = \{s_t \in \Pi \to I_t\}_{t \in T}$ と定める．
遷移関係 $\rho \in A(\Pi \cup \Pi')$
初期状態 初期状態を表現する式 $\Theta \in A(\Pi)$ と定める．

また，論理 $\mathcal{L}$ が区間論理であるとき，区間遷移システムとよぶ．

遷移システムによって得られる状態遷移から 1 つの列を計算と定めることで 1 つのプログラムの実行を表現できる．

定義 3.5.1.3 (計算)
遷移システム $\mathcal{A}_{\mathcal{L}} = (\Pi_{\mathcal{L}}, \Sigma_{\mathcal{L}}, \rho_{\mathcal{L}}, \Theta_{\mathcal{L}})$ について，以下の条件を満たす状態の列 $\sigma_{\mathcal{A}_{\mathcal{L}}} = s_0, s_1, \ldots$ のことを計算と定める．

(1) $s_0 \vDash \Theta_{\mathcal{L}}$
(2) $\forall i \in \mathbb{N}$ について $(s_i \triangleright s_{i+1}) \vDash \rho$

3.5.2 遷移システムと区間遷移システムの間の関係

区間遷移システムの計算が 1 つの遷移システムの計算を模倣する関係を定式化する．複数の実行環境における 1 つのプログラムの実行表現が，1 つの実行環境における 1 つのプログラムの実行表現を模倣することを表現する．

定義 3.5.2.1 (区間表現模倣性)
論理 $\mathcal{L}$ 上の遷移システム $\mathcal{A}_{\mathcal{L}}$ の計算 $\sigma_{\mathcal{L}}$ と区間論理 $I\mathcal{L}$ 上の区間遷移システム $\mathcal{A}_{I\mathcal{L}}$ 計算 $\sigma_{I\mathcal{L}}$ について以下の条件を満すとき，区間遷移システム $\mathcal{A}_{I\mathcal{L}}$ は遷移システム $A_{\mathcal{L}}$ を模倣すると定める．

- $\Pi_{\mathcal{A}_{\mathcal{L}}} = \Pi_{\mathcal{A}_{I\mathcal{L}}}$
- $\rho_{\mathcal{A}_{\mathcal{L}}} = \rho_{\mathcal{A}_{I\mathcal{L}}}$
- $\forall p \in \Pi\ [\![p]\!]_{(s_{0_{\mathcal{A}_{\mathcal{L}}}})} \in [\![p]\!]_{(s_{0_{\mathcal{A}_{I\mathcal{L}}}})}$
- $\forall i \in \mathbb{N} \forall p \in \Pi\ [\![p]\!]_{(s_{i_{\mathcal{A}_{\mathcal{L}}}} \triangleright s_{i+1_{\mathcal{A}_{\mathcal{L}}}})} \in [\![p]\!]_{(s_{i_{\mathcal{A}_{I\mathcal{L}}}} \triangleright s_{i+1_{\mathcal{A}_{I\mathcal{L}}}})}$

論理集合 L の任意の論理 $\mathcal{L}$ の遷移システム $\mathcal{A}_{\mathcal{L}}$ の計算 $\sigma_{\mathcal{L}}$ が区間表現可能な区間論理 $I\mathcal{L}$ の区間遷移システム $\mathcal{A}_{I\mathcal{L}}$ の計算 $\sigma_{I\mathcal{L}}$ で模倣できることを示す．区間遷移システムが複数の遷移システムの計算を一度に模倣できることを表現する．

定理 3.5.2.1 (区間表現可能模倣性)
論理 $\mathcal{L} \in L$ 上の遷移システム $\mathcal{A}_{\mathcal{L}}$ と区間表現可能な区間論理 $I\mathcal{L}$ 上の区間遷移システム $\mathcal{A}_{I\mathcal{L}}$ について以下の性質を満たすとき，すべての論理 $\mathcal{L} \in L$ 上の遷移システム $\mathcal{A}_{\mathcal{L}}$ の計算 $\sigma_{\mathcal{L}}$ を模倣する区間遷移システム $\mathcal{A}_{I\mathcal{L}}$ の計算 $\sigma_{I\mathcal{L}}$ が存在する．

- $\Pi_{\mathcal{A}_{\mathcal{L}}} = \Pi_{\mathcal{A}_{I\mathcal{L}}}$
- $\rho_{\mathcal{A}_{\mathcal{L}}} = \rho_{\mathcal{A}_{I\mathcal{L}}}$
- $\Theta_{\mathcal{A}_{\mathcal{L}}} = \Theta_{\mathcal{A}_{I\mathcal{L}}}$

証明.
論理 $\mathcal{L}$ とその上の遷移システムと計算 $\sigma_{\mathcal{A}_{\mathcal{L}}}$ をとる．区間論理 $I\mathcal{L}$ とその上の区間遷移システム $\mathcal{A}_{I\mathcal{L}}$ の計算を以下の手順で選ぶと条件を満たす計算 $\sigma_{\mathcal{A}_{I\mathcal{L}}}$ が得られる．

(1) $\sigma_{0_{\mathcal{A}_{I\mathcal{L}}}} \vDash \Theta_{\mathcal{A}_{\mathcal{L}}}$ このとき，区間表現可能な区間論理の構成 (定理. 3.4.3.2, 20 ページ) より $\forall p \in \Pi\ [\![p]\!]_{(s_{0_{\mathcal{A}_{\mathcal{L}}}})} \in [\![p]\!]_{(s_{0_{\mathcal{A}_{I\mathcal{L}}}})}$

(2) $j \in \mathbb{N}$ について，$\forall p \in \Pi\ [\![p]\!]_{(s_{j_{\mathcal{A}_{\mathcal{L}}}} \triangleright s_{j+1_{\mathcal{A}_{\mathcal{L}}}})} \in [\![p]\!]_{(s_{j_{\mathcal{A}_{I\mathcal{L}}}} \triangleright s_{j+1_{\mathcal{A}_{I\mathcal{L}}}})}$ が成り立つと仮定する．
   - $\forall p \in \Pi\ [\![p]\!]_{(s_{j+1_{\mathcal{A}_{I\mathcal{L}}}} \triangleright s_{\mathcal{A}_{I\mathcal{L}}})}$ を満たす $s_{\mathcal{A}_{I\mathcal{L}}}$ が 1 通りのとき，$j = i$ について，$\forall p \in \Pi\ [\![p]\!]_{(s_{j+1_{\mathcal{A}_{\mathcal{L}}}} \triangleright s_{\mathcal{A}_{\mathcal{L}}})} \in [\![p]\!]_{(s_{j+1_{\mathcal{A}_{I\mathcal{L}}}} \triangleright s_{\mathcal{A}_{I\mathcal{L}}})}$ を満たすから，そのような $s_{\mathcal{A}_{I\mathcal{L}}}$ を $s_{j+2_{\mathcal{A}_{I\mathcal{L}}}}$ として与える．
   - $\forall p \in \Pi\ [\![p]\!]_{(s_{j+1_{\mathcal{A}_{I\mathcal{L}}}} \triangleright s_{\mathcal{A}_{I\mathcal{L}}})}$ を満たす $s_{\mathcal{A}_{I\mathcal{L}}}$ が 2 通りのとき，$[\![l]\!]_{(s_{j+1_{\mathcal{A}_{\mathcal{L}}}} \triangleright s_{\mathcal{A}_{\mathcal{L}}})} \in [\![l]\!]_{(s_{j+1_{\mathcal{A}_{I\mathcal{L}}}} \triangleright s_{\mathcal{A}_{I\mathcal{L}}})}$ を満たす $s_{\mathcal{A}_{\mathcal{L}}}$) は $\forall p \in \Pi\ [\![p]\!]_{(s_{j+1_{\mathcal{A}_{\mathcal{L}}}} \triangleright s_{\mathcal{A}_{\mathcal{L}}})} \in [\![p]\!]_{(s_{j+1_{\mathcal{A}_{I\mathcal{L}}}} \triangleright s_{\mathcal{A}_{I\mathcal{L}}})}$ を満たすから，そのような $s_{\mathcal{A}_{I\mathcal{L}}}$ を $s_{j+2_{\mathcal{A}_{I\mathcal{L}}}}$ として与える．

第 4 章

結論と今後の課題

4.1 結論

プログラムの解析のためにプログラムを定式化するが，先行研究では解析する目的によっては，各実行環境に応じて定式化する必要があり，状況によっては定式化ができない場合がある．

本研究では，採用する論理における構文に 2 つの論理のためのソートを含むようなソートをもつよう制限を与え，記号の解釈として区間を用いた区間論理を定義した．また，論理と区間論理の間の関係を定めて具体的な区間論理の構成を提案し，その上の遷移システムによる計算が複数の実行環境における計算を模倣することを示した．

4.2 今後の課題

本研究では 1 つのプログラムに対する定式化を与えているが，プログラム間の関係や等価性を議論していない．例えば，ある 1 つのみの入力の計算パスが一致して 10 進数上位 3 桁のみが一致することを要求される時，古い区間遷移システムと新しい区間遷移システムと以下のような手順で等価性を判定だろう．

- 初期条件が一致する時，計算パスが一致する
- 古い区間遷移システムと新しい区間遷移システムの終了状態の最小値と最大値の 10 進数上位 3 桁が一致する

このような具体的な等価性の定式化を与える必要がある．

また，区間論理における関数記号の解釈として，想定する全ての実行環境における関数の適用結果の最小値と最大値を含むように与えていた．実行環境が再現できない実行環境の内，仕様が失われたものに関しては実験して仕様を確認する必要がある．論理集合の関数記号における性質によっては仕様の確認が不要である場合があるかもしれない．本研究では論理集合の全ての論理における対象集合に全順序が定義されていることを制限しており，順序がない対象集合における性質を考察していない．そのため，区間論理の各ソートの性質を検討することが課題である．

謝辞

本研究を進める上で，日頃から時間を割いて指導，議論して頂いた山田俊行講師，河内亮周教授，高木一義教授，多くの助言をくださり相談を受けてくださったコンピュータソフトウェア研究室の皆様，研究活動で困り事を解決してくださった落合美子事務員と森岡幸音事務員に感謝いたします．

参考文献

[Gal15] Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Dover, 2015, pp. 431–459.

[MP92] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems · Specification ·*. Springer-Verlag New York, 1992.

[Nec00] George C. Necula. "Translation Validation for an Optimizing Compiler". In: *Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation*. Association for Computing Machinery, 2000, pp. 83–94. URL: https://dl.acm.org/doi/pdf/10.1145/358438.349314.

[Nic07] D Nicolae. "Interval Analysis - A Powerfull Trend in Numerical Analysis". In: International Conference Trends and Challenges in Applied Mathematics. 2007. URL: https://www.researchgate.net/publication/236341177_Interval_Analysis_-_A_Powerfull_Trend_in_Numerical_Analysis.

[NNH05] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer, 2005.

[PSS98] A. Pnueli, M. Siegel, and E. Singerman. "Translation validation". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Springer Berlin Heidelberg, 1998, pp. 151–166. URL: https://link.springer.com/content/pdf/10.1007%5C%2FBFb0054170.pdf.

[SMK13] Thomas Arthur Leck Sewell, Magnus O. Myreen, and Gerwin Klein. "Translation validation for a verified OS kernel". In: *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2013, pp. 471–482. URL: https://dl.acm.org/doi/proceedings/10.1145/2491956.2462183.

[日本規 03] 日本規格協会. **プログラム言語Ｃ**. JISX3010. 2003. URL: https://www.jisc.go.jp/app/jis/general/GnrJISNumberNameSearchList?show&jisStdNo=X3010.

[日本規 11] 日本規格協会. **電子計算機プログラム言語ＣＯＢＯＬ**. JISX3002. 2011. URL: https://www.jisc.go.jp/app/jis/general/GnrJISNumberNameSearchList?show&jisStdNo=X3002.