顕在的契約計算におけるアップキャスト除去

関山 太朗, 五十嵐 淳

京都大学大学院情報学研究科

{t-sekiym,igarashi}@kuis.kyoto-u.ac.jp

概 要　顕在的契約計算体系は，実行時に検査されるソフトウェア契約の情報—例えば，スタックの pop 操作は非空スタックを引数とする，といった，単純な型では捉えられない制約条件—が，篩 (ふるい) 型 (refinement type) として静的型情報に明示的に現れている型付計算体系である．顕在的契約計算体系での実行時検査は型変換 (キャスト) により実現される．

本研究では多相関数型と再帰関数を含む顕在的契約計算体系 $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ において，部分型関係にある型の間のキャストであるアップキャストが除去可能であることを，$\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ の弱型付文脈等価性とそれに対し健全な論理関係を与え，アップキャストと恒等関数が論理関係にあることを証明することで示す．

1　はじめに

プログラムの最適化やリファクタリングでは，変更前後のプログラム片同士の振る舞いが「等しい」ことが求められる．文脈等価性 [10] はプログラム片の「等しさ」を表わす関係の一種で，どのようなプログラム文脈でも同じように振る舞うプログラム片の関係を表わしている．最適化などの変更が適用される前後のプログラム片が文脈等価であれば，プログラム全体の振る舞いを変えることなく，それらを交換することができる．しかし二つのプログラム片が文脈等価であることを示すためには，それらに対する「任意」のプログラム文脈を考慮しなくてはならず，直接プログラム片の文脈等価性を示すことは困難である．そこで文脈等価性を示す技法として，静的に型付けされるプログラミング言語では論理関係 [12] という型について帰納的に定義されたプログラム片の二項関係が用いられる．文脈等価性が論理関係を包含することが証明できれば，論理関係を用いて比較的容易にプログラム片が文脈等価であることを示すことができる．

顕在的契約計算体系 λ^H [6, 8] は，実行時に検査されるソフトウェア契約の情報—例えば，スタックの pop 操作は非空スタックを引数とする，といった，単純な型では捉えられない制約条件—が，篩 (ふるい) 型 (refinement type) として静的型情報に明示的に現れている型付計算体系である．顕在的契約計算体系での実行時検査は型変換 (キャスト) により実現される．λ^H に多相関数型を加えた計算体系 F_{H} [1] では部分型関係 $T_1 <: T_2$ が成り立つとき，T_1 の項を T_2 として扱うためには T_1 から T_2 へのキャストを挿入する必要があるが，部分型関係が成り立つような T_1 から T_2 へのキャストによる実行時検査は必ず成功すると考えられる．

そこで本研究では，多相関数型と再帰関数を含む顕在的契約計算体系 $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ における最適化として，部分型関係 $T_1 <: T_2$ が成り立つような型 T_1 から型 T_2 へのキャストであるアップキャストが除去可能であることを，アップキャストを恒等関数[1] で置き換えてもプログラム全体の振る舞いが変わらないことを証明することで示す．二つのプログラム片を交換してもプログラム全体の振る舞いが変化しないことを示すには，それらが文脈等価であることを示せば十分である．一般に静的型付言語では文脈等価なプログラム片には同じ型が付くことが期待されるが，$\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ で文脈等価性を示

[1] $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ では $\mathtt{fun}\ x : T_1.\ x$ に対して与えられる型は $T_1 \to T_1$ だけであるため，本稿ではこのような関数を恒等関数と呼ぶが，意味としては包含写像と考えることができる．

したいアップキャストと恒等関数に同じ型を付けることはできない．そのため本研究では，関係付けられた二つのプログラム片が任意のプログラム文脈の下で同じように振る舞い，かつ一方のプログラム片は必ずしも型付け可能でなくてもよい弱型付文脈等価性[2] という関係を用いる．本研究では，再帰関数を含むような計算体系で文脈等価性に対して健全な論理関係を与える手法である ⊤⊤ 閉包 [11] を用いて弱型付文脈等価性に対して健全な論理関係を定義し，アップキャストと恒等関数が論理関係にあることを示す．以下 $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ に関しては，弱型付文脈等価性を単に文脈等価性という．

本稿の構成は以下の通りである．2 節では本研究で取り扱う計算体系 $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ を導入する．3 節では $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ の文脈等価性について説明し，4 節では $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ の論理関係とその性質について述べる．5 節では論理関係を用いてアップキャストが除去できることを示し，6 節では関連研究について言及する．最後に 7 節では本研究の成果をまとめる．

本稿では集合 X の冪集合を $\mathcal{P}(X)$，写像 f の定義域を $\mathrm{dom}(f)$ と表わす．また写像 f，g に対する合成を $f \circ g$ とし，写像 $f\,g$ を次のように定義する．

$$f\,g(x) = \begin{cases} g(x) & (x \in \mathrm{dom}(g) \text{ の場合}) \\ f(x) & (x \notin \mathrm{dom}(g) \text{ の場合}) \end{cases}$$

本稿では紙面の都合上いくつかの定義を省略している．厳密な定義を載せたフルバージョンは以下の URL から取得できる．

`http://www.sato.kuis.kyoto-u.ac.jp/~t-sekiym/papers/fh-upcast/full.pdf`

2 顕在的契約計算 $\mathbf{F}_{\mathbf{H}}^{\mathbf{fix}}$

2.1 概略

計算体系 $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ は顕在的契約計算体系 λ^H [6, 8] へ多相関数型を加えた計算体系 F_{H} [1] に対し，さらに再帰関数を導入した計算体系である．

λ^H は型システムによる静的検査と，型変換 (キャスト) による実行時検査を備えた計算体系で，実行時に検査されるソフトウェア契約の情報を篩型を用いて静的型情報として表現することができる．篩型 $\{x{:}T \mid e\}$ は変数 x，型 T，真偽値型の項 e から構成される型で，型 T の値 v のうち，$e\,[v/x]$ が真に評価されるようなものの集合を表わす型である．例えば整数型 Int と Int 上の比較演算子 $<$ を用いると，正整数を表わす型は $\{x{:}\mathsf{Int} \mid 0 < x\}$ と書ける．また λ^H には依存関数型がある．依存関数型 $x{:}T_1 \to T_2$ は $x{:}T_1 \to T_2$ の値が T_1 の値 v に適用されると $T_2\,[v/x]$ の値を返す，ということを表わす型である．依存関数型を用いると，例えば整数に適用されるとそれより大きな整数を返す関数の型を $x{:}\mathsf{Int} \to \{y{:}\mathsf{Int} \mid x < y\}$ と書くことができる．

項が篩型の表明を満たすことは，キャストを用いて確認する．キャスト $\langle\ T_1 \Rightarrow T_2\ \rangle^l$ は T_1 の値に適用されると，その値が T_2 として振る舞うことができるかを検査する．検査に成功すれば値が返り，失敗すると例外 $\Uparrow l$ が発生する．l はキャストごとに付与されるラベルで，各キャストにはそれぞれ異なるラベルが与えられる．失敗したキャストは l によって識別することができる．例として，整数型から正整数型へのキャスト $\langle\ \mathsf{Int} \Rightarrow \{x{:}\mathsf{Int} \mid 0 < x\}\ \rangle^l$ について考える．整数 5 は $0 < 5$ が成り立つため，5 は正整数型として振る舞うことができる．すなわち，

$$\langle\ \mathsf{Int} \Rightarrow \{x{:}\mathsf{Int} \mid 0 < x\}\ \rangle^l\, 5 \longrightarrow^* 5$$

となる．一方，整数 0 に対しては $0 < 0$ は成り立たないため，正整数型へのキャストは失敗し，例外 $\Uparrow l$ が発生する．

$$\langle\ \mathsf{Int} \Rightarrow \{x{:}\mathsf{Int} \mid 0 < x\}\ \rangle^l\, 0 \longrightarrow^* \Uparrow l$$

[2] 弱型付文脈等価性は対称性が成り立たないため擬順序だが，型付け可能なプログラム片に限れば同値関係となる．

	$\simeq$	$\approx$	$\simeq \subseteq \approx$	$\langle\ T_1\ \Rightarrow\ T_2\ \rangle^l \simeq \mathtt{fun}\ x : T_1.\ x$
$\mathrm{F_H}$	◯			◯
$\mathrm{F_H^{fix}}$	◯	◯	◯	◯

表 1. Belo ら [1] による研究結果との比較

また，依存関数型や篩型間のキャストを行うことも可能である [6, 8, 1].

$\mathrm{F_H}$ は λ^H に多相関数型を加えた計算体系である．多相関数型を用いると存在型と依存和型を実現することができ，存在型と依存和型を組み合わせることでモジュールの仕様を篩型の表明として静的型情報に埋め込むことが可能となる．これにより，例えばデータ構造のスタックを表わすモジュールでは，pop 操作は非空のスタックにだけ適用可能であることを表現することができる．また λ^H では基本型に対してだけ篩型を定義することができたのに対し，$\mathrm{F_H}$ では型変数や依存関数型などの任意の型を用いて篩型を定義することができる．例えば λ^H で定義することができなかった $\{x{:}\alpha \,|\, \mathsf{true}\}$ や $\{f{:}(x{:}\mathsf{Int} \to \mathsf{Bool}) \,|\, f\,0\}$ などの篩型は $\mathrm{F_H}$ では有効な型である．

$\mathrm{F_H}$ では必ず停止するプログラムしか書くことができないが，$\mathrm{F_H^{fix}}$ では再帰関数を導入したことにより，例えばリストの長さを求めるような関数 $\mathsf{length} : \forall\,\alpha.\ \alpha\ \mathsf{List} \to \mathsf{Int}$ を記述することができ，それを用いるとリストの結合操作 append が返すリストの長さが結合対象のリストの長さの和に等しいといったような契約情報を，次のように表現することが可能となる．

$$\mathsf{append} : \forall\,\alpha.\ x{:}\alpha\ \mathsf{List} \to y{:}\alpha\ \mathsf{List} \to \{z{:}\alpha\ \mathsf{List} \,|\, \mathsf{length}\,\alpha\,z = \mathsf{length}\,\alpha\,x + \mathsf{length}\,\alpha\,y\}$$

λ^H の型付け規則には，部分型関係 $T_1 <: T_2$ が成り立つとき T_1 の項を T_2 の項として扱うことを許す規則 (subsumption rule) がある．このため例えば，型 $x{:}T_1 \to T_2$ の関数抽象 f を型 T_3 の値 v に適用する場合，T_3 が T_1 の部分型であれば v は T_1 の値と見なすことができるため，関数適用 $f\,v$ に型を付けることができる．一方 $\mathrm{F_H}$ では計算体系やメタ理論を簡潔にするために型付け規則に subsumption rule を含んでいない．そのため f と v による関数適用に型を付けるためにはアップキャストを用いて $f\,(\langle\ T_3\ \Rightarrow\ T_1\ \rangle^l\,v)$ と書かなければならず，v に対して必ず成功する実行時検査を行うことになる．Belo らはこのようなアップキャストの挿入がプログラムの振る舞いに影響しないことを，アップキャストと恒等関数が文脈等価であることを証明することで示そうとした．具体的には $\mathrm{F_H}$ に対する論理関係を定義し，アップキャストと恒等関数が論理関係にあることを示した．しかし Belo らは論理関係と文脈等価性との関係について厳密に議論しておらず，また Belo らが定義した論理関係が文脈等価性に対して健全であることを示すことは困難であることがわかった．Knowles ら [8] も λ^H においてアップキャストを挿入してもプログラム全体の振る舞いが変わらないことを示すために，文脈等価性に対して健全な論理関係を与えようとしたが，健全性の証明が不完全であることが筆者らと Knowles らとの個人的なやりとりによって確認された．

本研究では $\mathrm{F_H^{fix}}$ の文脈等価性 $\approx$ とそれに対し健全な論理関係 $\simeq$ を与え，その上で論理関係にアップキャストと恒等関数が含まれることを示すことにより，アップキャストを恒等関数に置き換えてもプログラム全体の振る舞いが変わらないことを示す．本研究と，Belo らによって $\mathrm{F_H}$ について示された結果との比較を表 1 に示す．

2.2 構文

$\mathrm{F_H^{fix}}$ の構文を図 1 に示す．項，値，型を表わすメタ変数としてそれぞれ e，v，T を用いる．

基本型 B は真偽値型 Bool を含み，B の定数の集合を $\mathcal{K}_B$ とする．型変数は α を用いて表わす．依存関数型 $x{:}T_1 \to T_2$ は型 T_1 から型 T_2 への関数の型で，x は T_2 上で束縛されている型 T_1 の変数である．T_2 に x が自由に出現しないとき，単に $T_1 \to T_2$ と書く．$\forall\,\alpha.\ T$ は多相関数型で，型変数 α は T で束縛されている．篩型 $\{x{:}T \,|\, e\}$ の項 e は型 T の変数 x が自由に出現することのできる真偽

型

B ::= $\mathsf{Bool} \mid ...$

T ::= $B \mid \alpha \mid x{:}T_1 \to T_2 \mid \forall\alpha.\ T \mid \{x{:}T \mid e\}$

型環境

Γ ::= $e \mid \Gamma, x{:}T \mid \Gamma, \alpha$

項

v ::= $x \mid k \mid \mathtt{fun}\ f(x:T_1).\ e:T_2 \mid \Lambda\alpha.\ v \mid \Lambda\alpha.\ \mathsf{let}\ x\ =\ v\ \alpha\ \mathsf{in}\ \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\ x \mid \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l$

e ::= $v \mid \mathrm{op}\,(v_1, .., v_n) \mid v_1\ v_2 \mid v\ T \mid \langle\{x{:}T_1 \mid e_1\}, e_2, v\rangle^l \mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \mid \Uparrow l$

評価文脈

E ::= $[\,] \mid E[\ \mathsf{let}\ x = [\,]\ \mathsf{in}\ e\ ] \mid E[\ \langle\{x{:}T \mid e\}, [\,], v\rangle^l\ ]$

図 1. $\mathrm{F}^{\mathrm{fix}}_{\mathrm{H}}$ の構文

値型の項で，表明という．また厳密な定義は省略するが，篩型の外側の表明を全て取り除いた型を返す写像を unref とする．例えば $\mathrm{unref}\,(\{x{:}\{y{:}(\mathsf{Bool} \to \{w{:}\mathsf{Bool} \mid e_3\}) \mid e_2\} \mid e_1\}) = \mathsf{Bool} \to \{w{:}\mathsf{Bool} \mid e_3\}$ となる．型環境 Γ は変数と型のペア $x : T$ と型変数 α の列で，変数や型変数は重複しないものとする．

変数は x，y，z で表わす．関数抽象 $\mathtt{fun}\ f(x : T_1).\ e : T_2$ は型 T_1 から型 T_2 への再帰関数で，x は引数に対応する変数，f は関数抽象自身を表わす変数で，ともに e 上で束縛される．再帰は，関数抽象が値に適用された際に，e 上の変数 f に関数抽象自身を代入することで実現される．f が e に自由に出現しないときや T_2 が明らかであるときなどは，$\mathtt{fun}\ x : T_1.\ e : T_2$ や $\mathtt{fun}\ x : T_1.\ e$ と書く．型抽象 $\Lambda\alpha.\ v$ の本体は値に制限されており，型抽象 $\Lambda\alpha.\ \mathsf{let}\ x\ =\ v\ \alpha\ \mathsf{in}\ \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\ x$ は多相関数型間のキャスト $\langle\ \forall\alpha.\ T_1\ \Rightarrow\ \forall\alpha.\ T_2\ \rangle^l$ を値 v に適用した際の簡約結果を表わしている．これらの型抽象の本体における制限は ML の値多相に対応している．let 式 $\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$ は変数 x，項 e_1，e_2 から構成され，x は e_2 上で束縛されている．

定数 k は基本型の値で，Bool の値 true，false を含む．op は各基本型に対する等号を含む組み込みの演算子で，op の意味は $[\![\mathsf{op}]\!]$ によって与えられる．k，op の型は閉じた型への写像 ty を用いて $\mathsf{ty}\,(k)$，$\mathsf{ty}\,(\mathrm{op})$ と書き，$\mathsf{ty}\,(\mathrm{op})$ は単相の依存関数型 $x_1 : T_1 \to .. \to x_n : T_n \to T$ に限定される．$\mathsf{ty}\,(k)$，$\mathsf{ty}\,(\mathrm{op}) = x_1 : T_1 \to .. \to x_n : T_n \to T$ については，それぞれ $\exists B.\ \mathrm{unref}\,(\mathsf{ty}\,(k)) = B$，$\exists B.\ \mathrm{unref}\,(T_i) = B$ が成り立つ．また k は $\mathsf{ty}\,(k)$ に出現する全ての表明を満たすような値で，$[\![\mathsf{op}]\!]$ は op の引数型をもつ閉じた値に対して，かつそのときのみ定義され，$[\![\mathsf{op}]\!]$ の戻り値には op の戻り値型を付けることができるものとする．

キャスト $\langle\ T_1\ \Rightarrow\ T_2\ \rangle^l$ は型 T_1 から型 T_2 への関数で，値 v に適用されると v が型 T_2 として振舞えるかを検査し，失敗すれば例外 $\Uparrow l$ が発生する．T_2 を篩型 $\{x{:}T \mid e\}$ とすると，v が T の値として振る舞うことができ，かつ $e\,[v/x]$ の評価結果が true になるとき，v は $\{x{:}T \mid e\}$ として振る舞うことができる．v が T として振る舞えるかはキャスト $\langle\ T_1\ \Rightarrow\ T\ \rangle^l$ によって確認することができる．また $e\,[v/x]$ の評価結果を検査するために，キャストの適用は表明検査 $\langle\{x{:}T \mid e\}, e_1, v\rangle^l$ に簡約される．e_1 は $e\,[v/x]$ もしくはそれを簡約した項で，e_1 が true になれば表明検査は v に評価され，false になれば例外 $\Uparrow l$ が発生する．表明検査中の篩型 は $\mathrm{F}^{\mathrm{fix}}_{\mathrm{H}}$ の型健全性を示すために必要となる．型抽象 $\Lambda\alpha.\ \mathsf{let}\ x\ =\ v\ \alpha\ \mathsf{in}\ \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\ x$，表明検査 $\langle\{x{:}T_1 \mid e_1\}, e_2, v\rangle^l$，例外 $\Uparrow l$ はソースプログラムには出現せず，項の評価時にのみ出現する項である．

項 e に自由に出現する変数 (または型変数) の集合を $\mathsf{FV}\,(e)$ (または $\mathsf{FTV}\,(e)$) とし，e に対する変数 (または型変数) $x_1, .., x_n$ (または $\alpha_1, .., \alpha_n$) へ $v_1, .., v_n$ (または型 $T_1, .., T_n$) を capture avoiding に代入して得られる項を $e\,[v_1/x_1, .., v_n/x_n]$ (または $e\,[T_1/\alpha_1, .., T_n/\alpha_n]$) と表わす．また，$\mathsf{FV}\,(e) = \emptyset$ かつ $\mathsf{FTV}\,(e) = \emptyset$ であるような項 e を閉じた項と呼び，α 変換に対して同値な項は同じ項とみなす．

型についても同様である．

評価文脈 [4] は項を構成する部分項をどのような順序で，もしくどの部分項を評価するのかを与えている．評価文脈の穴 ([]) は評価されるべき項が配置される場所を表わす．また値以外の部分項から構成されるような項は let 式と表明検査のみで，関数適用，型適用などの部分項は値だけに限定されている．二つ以上の部分項から構成される項の操作的意味論を定義するために，項の各構文要素についてそれぞれ評価文脈を用意する方法もあるが，本研究では評価文脈の種類を減らすために let 式を用いて部分項の評価順序を明示するようにしている．例えば一般的なラムダ計算における関数適用 $e_1\, e_2$ は $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ では $\mathsf{let}\ x = e_1\ \mathsf{in}\ \mathsf{let}\ y = e_2\ \mathsf{in}\ x\, y$ と表現される．

2.3 操作的意味論

値呼び出しによる $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ の操作的意味論を図 2 に示す．

簡約規則 $e_1 \rightsquigarrow e_2$ は項 e_1 を項 e_2 へ簡約する規則である．簡約規則については，関数適用を除いて F_{H} のものと同様であるため，詳しくは [1] を参照されたい．(R_Op) は op に対して適用される規則である．(R_Beta)(または (R_TBeta1)，(R_TBeta2)) は関数適用 (または型適用) で，関数抽象 (または型抽象) の本体に出現する変数 (または型変数) を値と関数抽象自身 (または型) で置換する．(R_Let) は let 式全体を簡約するための規則である．

(R_Refl)，(R_Fun)，(R_Forall) はそれぞれ同じ型，依存関数型，多相関数型の間のキャストを値に適用した際に用いる簡約規則で，(R_Check)，(R_Forget)，(R_PreCheck) はキャスト元もしくはキャスト先の型が篩型であるときに適用される．$\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ では篩型を用いて篩型をつくることができるため，表明検査を用いて実際の篩型の型検査を行うための規則 (R_Check) に加えて，キャスト元とキャスト先の篩型を一段階はぎ落とす (R_Forget)，(R_PreCheck) が必要となる．(R_OK)，(R_Fail) はそれぞれ型検査が成功，失敗した場合に適用される簡約規則である．

図 2 の $e_1 \longrightarrow e_2$ は項の評価規則で，$\longrightarrow$ の反射的推移的閉包を $\longrightarrow^*$ とする．(E_Red) は評価規則が簡約規則を包含することを示しており，(E_Blame) は例外が発生した際に適用される評価規則で，例外が発生すると項全体がただちに例外へ評価されることを表わしている．(E_Compat) は部分項を簡約するための規則である．

図 2 の最後の三つの関係 $e\downarrow$，$e \Uparrow l$，$e\upharpoonright$ は評価結果を表わしている．$e\downarrow$ は e が値に評価されたことを示しており，$e \Uparrow l$ は e が例外 $\Uparrow l$ に評価されたことを表わしている．また $e\upharpoonright$ は e が stuck 状態に，すなわち e の評価が値でも例外でもない状態で停止したことを示す関係である．

2.4 型システム

$\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ の型システムは相互再帰的に定義された三つの判断 $\vdash \Gamma$，$\Gamma \vdash T$，$\Gamma \vdash e : T$ から構成される．$\vdash \Gamma$ は型環境 Γ が well-formed であることを示す判断で，$\Gamma \vdash T$ は型環境 Γ の下で型 T が well-formed であることを示す判断である．また $\Gamma \vdash e : T$ は型環境 Γ の下で項 e に型 T が付くことを示す判断である．これらの判断を導出する推論規則を図 3 に示す．推論規則の多くは F_{H} のものと同様であるため，詳しくは [1] を参照されたい．以下では特に注意すべき規則についてだけ説明する．

(WF_Refine) は篩型 $\{x{:}T \mid e\}$ が well-formed であることを導出するための規則である．$\{x{:}T \mid e\}$ が well-formed であるためには，それを構成する型 T が well-formed でなければならず，また e は型 T の変数 x が自由に出現する真偽値型の項なので，e は $x : T$ を加えた型環境の下で Bool 型を付けられなければならない．

(T_Blame) は例外に対する型付け規則である．例外は項の評価中に発生するため，その型は閉じていなくてならない．(T_Abs) は関数抽象に対して適用される型付け規則で，関数抽象の本体は引数 x と関数抽象自身を表わす変数 f を加えた型環境の下で型付けされる．また関数抽象に付く型は依存関数型で，T_2 には f は出現しない．(T_Let) は let 式のための型付け規則である．e_2 の

$\boxed{e_1 \rightsquigarrow e_2}$

$$
\begin{array}{rcll}
\mathrm{op}\,(v_1, .., v_n) & \rightsquigarrow & [\![\mathsf{op}]\!]\,(v_1, .., v_n) & (\text{R\_Op}) \\
(\mathtt{fun}\, f(x : T_1).\ e : T_2)\, v & \rightsquigarrow & e\,[\mathtt{fun}\, f(x : T_1).\ e : T_2/f, v/x] & (\text{R\_Beta}) \\
(\Lambda\alpha.\ v)\, T & \rightsquigarrow & v\,[T/\alpha] & (\text{R\_TBeta1}) \\
(\Lambda\alpha.\ \mathsf{let}\ x\ =\ v\ \alpha\ \mathsf{in}\ \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\ x)\, T & \rightsquigarrow & (\mathsf{let}\ x = v\,\alpha\ \mathsf{in}\ \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\, x)\,[T/\alpha] & (\text{R\_TBeta2}) \\
\langle\ T\ \Rightarrow\ T\ \rangle^l\, v & \rightsquigarrow & v & (\text{R\_Refl}) \\
\langle\ x{:}T_{11} \to T_{12}\ \Rightarrow\ y{:}T_{21} \to T_{22}\ \rangle^l\, v & \rightsquigarrow & & (\text{R\_Fun}) \\
\multicolumn{4}{r}{\mathtt{fun}\ y : T_{21}.\ \mathsf{let}\ x = \langle\ T_{21}\ \Rightarrow\ T_{11}\ \rangle^l\, y\ \mathsf{in}\ \mathsf{let}\ z = v\, x\ \mathsf{in}\ \langle\ T_{12}\ \Rightarrow\ T_{22}\ \rangle^l\, z : T_{22}} \\
\multicolumn{4}{r}{(\text{ ただし } x \neq y \text{ かつ } z \text{ is a fresh variable})} \\
\langle\ \forall\alpha.\ T_1\ \Rightarrow\ \forall\alpha.\ T_2\ \rangle^l\, v & \rightsquigarrow & \Lambda\alpha.\ \mathsf{let}\ x\ =\ v\ \alpha\ \mathsf{in}\ \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\ x & (\text{R\_Forall}) \\
\multicolumn{4}{r}{(\text{ ただし } \forall\alpha.\ T_1 \neq \forall\alpha.\ T_2 \text{ かつ } x \text{ is a fresh variable})} \\
\langle\ T\ \Rightarrow\ \{x{:}T \mid e\}\ \rangle^l\, v & \rightsquigarrow & \langle\{x{:}T \mid e\}, e\,[v/x], v\rangle^l & (\text{R\_Check}) \\
\langle\ \{x{:}T_1 \mid e_1\}\ \Rightarrow\ T_2\ \rangle^l\, v & \rightsquigarrow & \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\, v & (\text{R\_Forget}) \\
\multicolumn{4}{r}{(\text{ ただし } T_2 \neq \{x{:}T_1 \mid e_1\} \text{ かつ } T_2 \neq \{y{:}\{x{:}T_1 \mid e_1\} \mid e_2\})} \\
\langle\ T_1\ \Rightarrow\ \{x{:}T_2 \mid e_2\}\ \rangle^l\, v & \rightsquigarrow & & (\text{R\_PreCheck}) \\
\multicolumn{4}{r}{\mathsf{let}\ y = \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\, v\ \mathsf{in}\ \langle\ T_2\ \Rightarrow\ \{x{:}T_2 \mid e_2\}\ \rangle^l\, y} \\
\multicolumn{4}{r}{(\text{ ただし } T_1 \text{ は篩型ではなく}, T_1 \neq T_2 \text{ かつ } y \text{ is a fresh variable})} \\
\langle\{x{:}T \mid e\}, \mathsf{true}, v\rangle^l & \rightsquigarrow & v & (\text{R\_OK}) \\
\langle\{x{:}T \mid e\}, \mathsf{false}, v\rangle^l & \rightsquigarrow & \Uparrow l & (\text{R\_Fail}) \\
\mathsf{let}\ x = v\ \mathsf{in}\ e & \rightsquigarrow & e\,[v/x] & (\text{R\_Let})
\end{array}
$$

$\boxed{e_1 \longrightarrow e_2}$

$$
\frac{e_1 \rightsquigarrow e_2}{e_1 \longrightarrow e_2}\ \text{E\_Red} \qquad \frac{E \neq [\,]}{E[\ \Uparrow l\ ] \longrightarrow \Uparrow l}\ \text{E\_Blame} \qquad \frac{e_1 \longrightarrow e_2 \quad \forall l.\ e_2 \neq \Uparrow l}{E[\ e_1\ ] \longrightarrow E[\ e_2\ ]}\ \text{E\_Compat}
$$

$\boxed{e\downarrow}$ $\boxed{e \Uparrow l}$ $\boxed{e\uparrow}$

$$
\frac{e \longrightarrow^* v}{e\downarrow}\ \text{Term\_Val} \qquad \frac{e \longrightarrow^* \Uparrow l}{e \Uparrow l}\ \text{Term\_Blame}
$$

$$
\frac{e \longrightarrow^* e' \quad e' \not\longrightarrow \quad \forall v.\ e' \neq v \quad \forall l.\ e' \neq \Uparrow l}{e\uparrow}\ \text{Term\_Stuck}
$$

図 2. $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ の操作的意味論

型 T_2 には型 T_1 の変数 x が自由に出現することができる．let 式全体の型は T_2 上の x を e_1 で置換したものであるが，$\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ の項は項の代入について閉じていない．そのため (T_Let) では次のような型の構造に関して帰納的に定義された写像 ϕ を用いて，T_2 上の変数 x に e_1 を代入したものと同じような型を得る．

$$
\begin{array}{rcll}
\phi\,(\alpha, x, e) & = & \alpha & \\
\phi\,(B, x, e) & = & B & \\
\phi\,(y{:}T_1 \to T_2, x, e) & = & y{:}\phi\,(T_1, x, e) \to \phi\,(T_2, x, e) & (x \neq y \text{ かつ } y \notin \mathsf{FV}\,(e) \text{ の場合}) \\
\phi\,(\forall\alpha.\ T, x, e) & = & \forall\alpha.\ \phi\,(T, x, e) & (\alpha \notin \mathsf{FTV}\,(e)) \\
\phi\,(\{y{:}T' \mid e'\}, x, e) & = & \{y{:}\phi\,(T', x, e) \mid \mathsf{let}\ x = e\ \mathsf{in}\ e'\} & (x \neq y, x \in \mathsf{FV}\,(e') \text{ かつ } y \notin \mathsf{FV}\,(e) \text{ の場合}) \\
\phi\,(\{y{:}T' \mid e'\}, x, e) & = & \{y{:}\phi\,(T', x, e) \mid e'\} & (x \neq y \text{ かつ } x \notin \mathsf{FV}\,(e') \text{ の場合})
\end{array}
$$

$\boxed{\vdash \Gamma}$

$$\frac{}{\vdash \emptyset}\ \text{WF\_Empty} \qquad \frac{\vdash \Gamma \quad \Gamma \vdash T}{\vdash \Gamma, x{:}T}\ \text{WF\_ExtendVar} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \alpha}\ \text{WF\_ExtendTVar}$$

$\boxed{\Gamma \vdash T}$

$$\frac{\vdash \Gamma}{\Gamma \vdash B}\ \text{WF\_Base} \qquad \frac{\vdash \Gamma \quad \alpha \in \Gamma}{\Gamma \vdash \alpha}\ \text{WF\_TVar} \qquad \frac{\Gamma, \alpha \vdash T}{\Gamma \vdash \forall \alpha.\ T}\ \text{WF\_Forall}$$

$$\frac{\Gamma \vdash T_1 \quad \Gamma, x{:}T_1 \vdash T_2}{\Gamma \vdash x : T_1 \to T_2}\ \text{WF\_Fun} \qquad \frac{\Gamma \vdash T \quad \Gamma, x{:}T \vdash e : \mathsf{Bool}}{\Gamma \vdash \{x{:}T \mid e\}}\ \text{WF\_Refine}$$

$\boxed{\Gamma \vdash e : T}$

$$\frac{\vdash \Gamma \quad x{:}T \in \Gamma}{\Gamma \vdash x : T}\ \text{T\_Var} \qquad \frac{\vdash \Gamma}{\Gamma \vdash k : \mathsf{ty}(k)}\ \text{T\_Const} \qquad \frac{\vdash \Gamma \quad \emptyset \vdash T}{\Gamma \vdash \Uparrow l : T}\ \text{T\_Blame}$$

$$\frac{\begin{array}{l}\vdash \Gamma \quad \mathsf{ty}(\mathsf{op}) = x_1 : T_1 \to \ldots \to x_n : T_n \to T \\ \forall i \in \{1, .., n\}.\ \Gamma \vdash v_i : T_i[v_1/x_1, .., v_{i-1}/x_{i-1}]\end{array}}{\Gamma \vdash \mathsf{op}(v_1, .., v_n) : T[v_1/x_1, .., v_n/x_n]}\ \text{T\_Op}$$

$$\frac{\Gamma, x{:}T_1, f{:}(x{:}T_1 \to T_2) \vdash e : T_2 \quad f \notin \mathsf{FV}(T_2)}{\Gamma \vdash \mathtt{fun}\ f(x : T_1).\ e : T_2 : (x{:}T_1 \to T_2)}\ \text{T\_Abs}$$

$$\frac{\Gamma \vdash v_1 : (x{:}T_1 \to T_2) \quad \Gamma \vdash v_2 : T_1}{\Gamma \vdash v_1\, v_2 : T_2\,[v_2/x]}\ \text{T\_App} \qquad \frac{\Gamma \vdash e_1 : T_1 \quad \Gamma, x{:}T_1 \vdash e_2 : T_2}{\Gamma \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \phi(T_2, x, e_1)}\ \text{T\_Let}$$

$$\frac{\Gamma, \alpha \vdash v : T}{\Gamma \vdash \Lambda\alpha.\ v : \forall \alpha.\ T}\ \text{T\_TAbs1} \qquad \frac{\Gamma, \alpha \vdash \mathsf{let}\ x = v\,\alpha\ \mathsf{in}\ \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\, x : T}{\Gamma \vdash \Lambda\alpha.\ \mathsf{let}\ x\ =\ v\ \alpha\ \mathsf{in}\ \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l\ x : \forall \alpha.\ T}\ \text{T\_TAbs2}$$

$$\frac{\Gamma \vdash v : \forall \alpha.\ T_1 \quad \Gamma \vdash T_2}{\Gamma \vdash v\ T_2 : T_1\,[T_2/\alpha]}\ \text{T\_TApp} \qquad \frac{\vdash \Gamma \quad \emptyset \vdash e : T_1 \quad \emptyset \vdash T_2 \quad T_1 \equiv T_2}{\Gamma \vdash e : T_2}\ \text{T\_Conv}$$

$$\frac{\Gamma \vdash T_1 \quad \Gamma \vdash T_2 \quad T_1 \parallel T_2}{\Gamma \vdash \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l : T_1 \to T_2}\ \text{T\_Cast} \qquad \frac{\vdash \Gamma \quad \emptyset \vdash v : \{x{:}T \mid e\}}{\Gamma \vdash v : T}\ \text{T\_Forget}$$

$$\frac{\vdash \Gamma \quad \emptyset \vdash \{x{:}T_1 \mid e_1\} \quad \emptyset \vdash e_2 : \mathsf{Bool} \quad \emptyset \vdash v : T_1 \quad e_1\,[v/x] \longrightarrow^* e_2}{\Gamma \vdash \langle \{x{:}T_1 \mid e_1\}, e_2, v \rangle^l : \{x{:}T_1 \mid e_1\}}\ \text{T\_Check}$$

$$\frac{\vdash \Gamma \quad \emptyset \vdash v : T \quad \emptyset \vdash \{x{:}T \mid e\} \quad e\,[v/x] \longrightarrow^* \mathsf{true}}{\Gamma \vdash v : \{x{:}T \mid e\}}\ \text{T\_Exact}$$

図 3. $\mathrm{F}^{\mathrm{fix}}_{\mathrm{H}}$ の型付け規則

(T_Cast) はキャストに対して適用される規則である．キャスト中の型は共に well-formed でなければならず，また $T_1 \parallel T_2$ は，厳密な定義は省略するが，T_1，T_2 が同じ構造であることを表わしている．型が同じ構造をしているとは，それに出現する全ての篩型 $\{x{:}T \mid e\}$ を T に置き換えたような型が等しいことをいう．

(T_Conv)，(T_Forget)，(T_Check)，(T_Exact) は評価途中の項に型を付けるための規則

である．項の評価が進むことで，型中に出現する項が変化する場合がある．例として let 式 $\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$ を評価することを考える．このとき e_1 が e_1' に簡約されたとすると，let 式全体は $\mathsf{let}\ x = e_1'\ \mathsf{in}\ e_2$ へと簡約されたことになる．ここで簡約前の let 式の型を $\phi(\{y{:}B \mid e\}, x, e_1) = \{y{:}B \mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e\}$ とすると，簡約後の let 式の型は $\phi(\{y{:}B \mid e\}, x, e_1') = \{y{:}B \mid \mathsf{let}\ x = e_1'\ \mathsf{in}\ e\}$ となる．(T_Conv) はこのような簡約前後の項に同じ型を付けるために適用され，型 T_1 の値に，T_1 に出現する項を簡約したような型 T_2 を付けることができる型付け規則である．$\equiv$ は型上の並行簡約 $\Rrightarrow$ の推移的対称的閉包で，T_1 と T_2 に出現する表明が並行簡約によって関係付けられていることを表わしており，例えば並行簡約における篩型間の関係は，項に関する並行簡約関係 $e_1 \Rrightarrow e_2$ を用いると次のように定義される．

$$\frac{T_1 \Rrightarrow T_2 \quad e_1 \longrightarrow e_2}{\{x{:}T_1 \mid e_1\} \Rrightarrow \{x{:}T_2 \mid e_2\}}\ \text{EP\_TRefine}$$

(T_Forget) は篩型 $\{x{:}T \mid e\}$ が付けられる値は，その篩型を構成する型 T を付けることができることを表わす型付け規則である．(T_Check) は表明検査 $\langle\{x{:}T_1 \mid e_1\}, e_2, v\rangle^l$ に対する型付け規則で，e_2 は $e_1\,[v/x]$ を簡約した結果であることが求められる．$\langle\{x{:}T_1 \mid e_1\}, e_2, v\rangle^l$ は篩型へのキャストと値の適用 $\langle\ T_1 \Rightarrow \{x{:}T_1 \mid e_1\}\ \rangle^l\, v$ を簡約して得られるので，篩型が付けられる．(T_Exact) は表明検査 $\langle\{x{:}T_1 \mid e_1\}, e_2, v\rangle^l$ において検査が成功した際の簡約結果である v に篩型 $\{x{:}T_1 \mid e_1\}$ を付けるための規則である．

2.5 型健全性

Belo らは subject reduction と progress を経由して $\mathrm{F_H}$ の型健全性を示した [1] が，$\mathrm{F_H^{fix}}$ においても同様の手法で型健全性を示すことができる．

定理 2.1 (強型健全性)**.** $\emptyset \vdash e : T$ かつ e の簡約が停止するとき，次のいずれかが成り立つ．

1. $e \longrightarrow^* v$ かつ $\emptyset \vdash v : T$
2. $e \longrightarrow^* \Uparrow l$

3 弱型付文脈等価性

本研究では $\mathrm{F_H^{fix}}$ においてアップキャストが除去可能であることを示すために，アップキャストを恒等関数に置換しても，プログラム全体の振る舞いが変わらないことを示す．置換してもプログラム全体の振る舞いが変化しないことを示す関係として文脈等価性 [10] がある．一般に静的型付言語では文脈等価な二項には同じ型が付くことが期待されるが，$\mathrm{F_H^{fix}}$ では文脈等価性を示したいアップキャストと恒等関数には同じ型を付けることはできない．例えばアップキャスト $\langle\ T_1 \Rightarrow T_2\ \rangle^l$ の型は $T_1 \to T_2$ であるが，恒等関数 $\mathtt{fun}\ x : T_1.\ x$，$\mathtt{fun}\ x : T_2.\ x$ の型はそれぞれ $T_1 \to T_1$，$T_2 \to T_2$ であり，いずれもアップキャストの型とは異なる．そのため，本研究では関係付けられた二項は任意の文脈で同じように振る舞い，かつそのうち一方には必ずしも型が付かないような関係である弱型付文脈等価性 $\approx$ を用いて，アップキャスト除去が可能であることを示す．

二つの項が文脈等価であるかは，それらを任意の文脈で評価した結果を観測することで判別される．再帰関数を含むような計算体系では一般的に観測対象として停止性が用いられ，本研究においても項の停止性を観測する．また $\mathrm{F_H^{fix}}$ には例外が存在するため，項の評価が停止した場合その結果が値であるか例外であるかを観測する必要がある．さらに本研究では文脈等価な二つの項のうち，一方には必ずしも型が付かず，そのような項を評価して停止した場合，値でも例外でもない結果を得る可能性がある．よって $\mathrm{F_H^{fix}}$ における観測対象は，項が (1) 値に評価されたか，(2) 例外に評価されたか，(3) 値でも例外でもない項に評価されたかの三つとなる．

定義 3.1. e と e' が以下の条件を満たすとき，e と e' は振る舞いが等しいといい，$e \Downarrow e'$ と表わす．

1. $e\downarrow \iff e'\downarrow$
2. `<< no parses (char 8): forall l***. e | l iff e'| l >>`
3. $e\upharpoonright \iff e'\upharpoonright$

穴が空いた項 e に対し，その穴を項 e' で埋めることを $e[\ e'\ ]$ と書くとすると，二つの項 e，e' が文脈等価であるとは，一般に任意の穴の空いた項 e_1 に対し，$e_1[\ e\ ] \Downarrow e_1[\ e'\ ]$ が成り立つことと定義される．しかしこのような定義の場合，e，e' が型環境 Γ の下で型付けされるとき，e_1 は，その穴の位置における型環境が Γ に一致するものに限る，といったように開いた項に対する文脈等価性の定義が複雑になるなどの問題がある．そのため，弱型付文脈等価性は Pitts [11] や Lassen [9] のスタイルに従って，ある条件を満たす最大の集合と定義する．

定義 3.2 (Adequate)**.** X が，$(\emptyset, e, e', T) \in X$ であれば $e \Downarrow e'$ が成り立つような集合であるとき，X は *adequate* であるという．

定義 3.3 (弱型付文脈等価性)**.** 項に関する弱型付文脈等価性 $\approx$ は (Γ, e, e', T) もしくは (Γ, T, T') を要素とし，以下の条件を満たす最大の集合である．ただし，$(\Gamma, e, e', T) \in \approx$ を $\Gamma \vdash e \approx e' : T$，$(\Gamma, T, T') \in \approx$ を $\Gamma \vdash T \approx T' : *$ と書く．

1. $\Gamma \vdash e \approx e' : T$ のとき $\Gamma \vdash e : T$ が導出される．
2. $\Gamma \vdash T \approx T' : *$ のとき $\Gamma \vdash T$ が導出される．
3. $\approx$ は *adequate* である．
4. 反射性，*(*型付項に関する*)* 対称性，推移性，置換性 *(*値と型の代入について閉じているという性質*)*，互換性 *(*文脈について閉じているという性質*)* を満たす．互換性は各構文要素ごとに定義され，例えば関数抽象や関数適用に対する規則は

$$\frac{\Gamma \vdash T_{11} \approx T_{21} : * \quad \Gamma, x{:}T_{11} \vdash T_{12} \approx T_{22} : * \quad \Gamma, x{:}T_{11}, f{:}(x{:}T_{11} \to T_{12}) \vdash e_1 \approx e_2 : T_{12}}{\Gamma \vdash \mathtt{fun}\ f(x : T_{11}).\ e_1 : T_{12} \approx \mathtt{fun}\ f(x : T_{21}).\ e_2 : T_{22} : (x{:}T_{11} \to T_{12})}\ \text{CE\_Abs}$$

$$\frac{\Gamma \vdash v_{11} \approx v_{21} : (x{:}T_1 \to T_2) \quad \Gamma \vdash v_{12} \approx v_{22} : T_1}{\Gamma \vdash v_{11}\ v_{12} \approx v_{21}\ v_{22} : T_2\,[v_{12}/x]}\ \text{CE\_App}$$

となる．ただし，*(*CE_App*)* のように型上の変数に代入される値や項については，必ず型が付いているもの *((*CE_App*)* の場合は v_{12}*)* を選ぶ．定義の詳細は *[15]* を参照されたい．

文脈等価性をこのように定義する利点として，具体的な文脈を扱うことなく文脈等価性を定義できることや，ある関係が文脈等価性に対して健全であることを示すには，その関係が adequate で，かつ文脈等価性の置換性と互換性を満たすことを示すだけで十分であることが挙げられる．後者の利点は，次の文脈等価性の存在の証明から得られる．

定理 3.1 (弱型付文脈等価性の存在性)**.** 弱型付文脈等価性 $\approx$ は存在する．

証明. $\approx$ の存在性を示すには，(Γ, e, e', T) と (Γ, T, T') を要素とし，置換性と互換性を満たす adequate な集合族の和集合が $\approx$ と等しいことを示せば十分である．これは Pitts と同じ方法 [11] により証明できる． □

4　論理関係

二つの項が文脈等価であることを示すためには「任意」の文脈について，その振る舞いが等しいことを証明しなければならないが，これを直接証明することは難しく，例えば文脈等価性が簡約規則について閉じていることを示すことさえ困難である．そのため本研究では，論理関係 [12] という型について帰納的に定義された関係を用いる．そして論理関係が文脈等価性に対して健全であることを証明し，アップキャストと恒等関数が交換可能であることを示すことに利用する．ただし，本研究での論理関係が文脈等価性に対して完全であるかはまだわかっていない．Belo らと本研究の論理関係の主な違いは以下の通りである．

1. Belo らの論理関係にある二項はともに型が付いている必要がなかったのに対し，本研究では論理関係にある二項のうち一方の項には型が付き，もう一方の項には必ずしも型が付かないとした．
2. Belo らは項が論理関係にあることを，それらの評価して得られた値が論理関係にあると定義した．しかし再帰関数のある計算体系ではこのような論理関係は文脈等価性に対し健全性を示すことができない．本研究では項に対する論理関係を定義するために ⊤⊤ 閉包 [11] を用いた．

定義 4.1. $\mathtt{Typ}$ を閉じた型の集合とする．$T \in \mathtt{Typ}$ のとき，

- $\mathsf{Term}\,(T) = \{e \mid \emptyset \vdash e : T\}$
- $\mathsf{UTerm} = \{e \mid \mathsf{FV}\,(e) = \emptyset\}$
- $\mathsf{Val}\,(T) = \{v \mid v \in Term(T)\}$
- $\mathsf{UVal} = \{v \mid v \in \mathsf{UTerm}\}$
- $\mathsf{ECtx} = \{E \mid \mathsf{FV}\,(E) = \emptyset\}$

とする．また $T, T' \in \mathtt{Typ}$ のとき，

- $\mathsf{TRel}\,(T) = \mathcal{P}(\mathsf{Term}\,(T) \times \mathsf{UTerm})$
- $\mathsf{VRel}\,(T) = \mathcal{P}(\mathsf{Val}\,(T) \times \mathsf{UVal})$
- $\mathsf{ERel}\,(T) = \mathcal{P}(\mathsf{ECtx} \times \mathsf{ECtx})$

とし，$\mathsf{TRel}\,(T)$，$\mathsf{VRel}\,(T)$，$\mathsf{ERel}\,(T)$ の要素を項関係，値関係，評価文脈関係と呼ぶ．

$\mathsf{ERel}\,(T)$ は，項関係もしくは値関係により評価文脈関係を定義した際にその型を保存するために $\mathsf{ERel}\,(T)$ の定義自体には出現しない T を用いている．項関係や値関係から評価文脈関係，または評価文脈関係から項関係を定義する方法は次の通りである．

定義 4.2 (項関係・評価文脈関係の操作)**.** $r \in \mathsf{TRel}\,(T)$，$c \in \mathsf{ERel}\,(T)$ とすると，$r^{\mathcal{E}} \in \mathsf{ERel}\,(T)$，$c^{\mathcal{R}} \in \mathsf{TRel}\,(T)$，$r^{\mathcal{V}} \in \mathsf{VRel}\,(T)$ を次のように定義する．

- $(E, E') \in r^{\mathcal{E}}$ *iff* $\forall (e, e') \in r.\ E[\ e\ ] \Downarrow E'[\ e'\ ]$ かつ $\exists T_1 \in \mathtt{Typ}.\ \emptyset \vdash E[\ e\ ] : T_1$
- $(e, e') \in c^{\mathcal{R}}$ *iff* $\forall (E, E') \in c.\ E[\ e\ ] \Downarrow E'[\ e'\ ]$ かつ $\exists T_1 \in \mathtt{Typ}.\ \emptyset \vdash E[\ e\ ] : T_1$
- $(e, e') \in r^{\mathcal{V}}$ *iff* $(e, e') \in r$ かつ $(e, e') \in \mathsf{Val}\,(T) \times \mathsf{UVal}$

定義 4.3. $\Gamma \vdash T$ とする．また写像 θ，δ を

$$\theta = \{\alpha \mapsto r, T, T' \mid \alpha \in \Gamma, r \in \mathsf{TRel}\,(T), \emptyset \vdash T \text{ かつ } T' \in \mathtt{Typ}\}$$
$$\delta = \{x \mapsto v, v' \mid x{:}T \in \Gamma, v \in \mathsf{Val}\,(T) \text{ かつ } v' \in \mathsf{UVal}\}$$

とし，型代入 θ_i と値代入 δ_i $(i \in \{1, 2\})$ を

$$\theta_i = \bigcup_{\alpha \mapsto r, T_1, T_2 \in \theta} [T_i/\alpha] \qquad \delta_i = \bigcup_{x \mapsto v_1, v_2 \in \delta} [v_i/x]$$

とする．このとき図 4 の関係を用いて，項関係 $T(\theta, \delta) \in \mathsf{TRel}\,(\theta_1(\delta_1(T)))$ を次のように T について帰納的に定義する．

$\boxed{\mathsf{Id}_B \in \mathsf{VRel}\,(B)}$

$\mathsf{Id}_B = \{(k,k) \mid k \in \mathcal{K}_B\}$

$\boxed{\mathtt{fun}\ x:r.\ \Lambda(v,v').\ \mathbf{R}\,(v,v') \in \mathsf{VRel}\,(x{:}T_1 \to T_2)}$

$r \in \mathsf{TRel}\,(T_1)$ で，任意の $(v,v') \in r^{\mathcal{V}}$ について，$\mathbf{R}\,(v,v') \in \mathsf{TRel}\,(T_2\,[v/x])$ が成り立つ．このとき
$(v_1,v_1') \in \mathtt{fun}\ x:r.\ \Lambda(v,v').\ \mathbf{R}\,(v,v')$ iff $\forall (v_2,v_2') \in r^{\mathcal{V}}.\ (v_1\ v_2, v_1'\ v_2') \in \mathbf{R}\,(v_2,v_2')$

$\boxed{\Lambda(r,T_1,T_1').\ \mathbf{R}\,(r,T_1,T_1') \in \mathsf{VRel}\,(\forall\,\alpha.\ T)}$

任意の T_1, T_1', r について，$\emptyset \vdash T_1$ かつ $r \in \mathsf{TRel}\,(T_1)$ であれば $\mathbf{R}\,(r,T_1,T_1') \in \mathsf{TRel}\,(T\,[T_1/\alpha])$ が成り立つ．このとき
$(v,v') \in \Lambda(r,T_1,T_1').\ \mathbf{R}\,(r,T_1,T_1')$ iff
　　$\forall T_1, T_1', r.\ \emptyset \vdash T_1$ かつ $r \in \mathsf{TRel}\,(T_1)$ ならば $(v\ T_1, v'\ T_1') \in \mathbf{R}\,(r,T_1,T_1')$

$\boxed{\{x:r \mid e_1,e_2\} \in \mathsf{VRel}\,(\{x{:}T \mid e\})}$

$r \in \mathsf{TRel}\,(T)$，$x{:}T \vdash e_1 : \mathsf{Bool}$ かつ $\mathsf{FV}\,(e_2) \subseteq \{x\}$ が成り立つ．このとき
$(v,v') \in \{x:r \mid e_1,e_2\}$ iff
　　$(v,v') \in r, e_1\,[v/x] \longrightarrow^* \mathsf{true}$ かつ $e_2\,[v'/x] \longrightarrow^* \mathsf{true}$

図 4. 論理関係における値関係

- $\alpha(\theta,\delta) = r^{\mathcal{V}\mathcal{E}\mathcal{R}}$ *(ただし* $\alpha \mapsto r, T_1, T_1' \in \theta$*)*
- $B(\theta,\delta) = (\mathsf{Id}_B)^{\mathcal{E}\mathcal{R}}$
- $(x{:}T_1 \to T_2)(\theta,\delta) = (\mathtt{fun}\ x : T_1(\theta,\delta).\ \Lambda(v,v').\ T_2(\theta,\delta\{x \mapsto v,v'\}))^{\mathcal{E}\mathcal{R}}$
- $(\forall\,\alpha.\ T)(\theta,\delta) = (\Lambda(r,T_1,T_1').\ T(\theta\{\alpha \mapsto r,T_1,T_1'\},\delta))^{\mathcal{E}\mathcal{R}}$
- $\{x{:}T \mid e\}(\theta,\delta) = (\{x : T(\theta,\delta) \mid \theta_1(\delta_1(e)), \theta_2(\delta_2(e))\})^{\mathcal{E}\mathcal{R}}$ *(ただし* $x \notin dom(\delta)$*)*

図 4 は各型について同じように振る舞う値の関係を定義している．また定義 4.3 では，図 4 の値関係と写像 θ，δ を用いて，同じように振る舞う項の関係 $T(\theta,\delta)$ を定義している．θ は型変数 α からその解釈 r と閉じた型 T，T' への写像で，δ は変数 x から閉じた値 v，v' への写像である．

型変数 α については，α の解釈である項関係 r に含まれる任意の値同士が，α として同じように振る舞うと定義される．Id_B は，基本型 B の値はその値自身と同じように振る舞い，それ以外の値とは同じように振る舞うことはできないことを表わしている．多相関数型として同じように振る舞う値は，任意の型に適用した際の振る舞いが等しい．依存関数型については，値 v_1，v_1' を引数の型として振る舞いが等しいような値 v_2，v_2' に適用して作られた関数適用 $v_1\,v_2$，$v_1'\,v_2'$ が，戻り値の型として同じように振る舞うとき，v_1，v_1' は依存関数型として同じように振る舞う．

篩型 $\{x{:}T \mid e\}$ として振る舞いが等しい値は，篩型を構成する型 T として同じように振る舞い，かつその値を e に代入した項が true に評価されなければならないが，項を評価するためにはその項が閉じている必要がある．e の変数，型変数は多相関数型や依存関数型によって束縛されている場合がある．例えば e の変数 y が依存関数型 $y{:}T_1 \to \{x{:}T \mid e\}$ によって束縛されており，$y{:}T_1 \to \{x{:}T \mid e\}$ の値 v_1，v_1' が関係付けられていることを示すことを考える．このとき定義から，T_1 について関係付けられている任意の値 v_2，v_2' に対して $v_1\,v_2$ と $v_1'\,v_2'$ がある型 T' について関係付けられていること示せば十分である．ここで問題となるのが T' の選び方である．一般に v_2 と v_2' は異なるため，

$v_1\,v_2$ と $v_1'\,v_2'$ が $\{x{:}T \mid e\}\,[v_2/y]$ もしくは $\{x{:}T \mid e\}\,[v_2'/y]$ として同じように振る舞うことは明らかではない．そこで本研究では Belo らと同様 [1] に，実際に閉じた項が必要になって初めて置換を行う方法を採用し，そのために変数から閉じた値への写像 δ を用いる．同様の理由により，型変数から閉じた型への写像である θ を用いる．δ，θ はそれぞれ依存関数型，多相関数型に関する項関係を定義する際に拡張される．

定義 4.3 では，図 4 の値関係に操作 $\mathcal{E}$，$\mathcal{R}$ を適用することで，同じように振る舞う項関係を定義している．評価文脈関係 $r^{\mathcal{E}}$ に含まれる評価文脈の組 (E, E') は，r に含まれる任意の値の組 (v, v') に対して $E[\,v\,] \Downarrow E'[\,v'\,]$ となる．ここで，$(e, e') \in r^{\mathcal{ER}}$ であるような任意の項 e，e' が値に評価される場合，$r^{\mathcal{ER}}$ の定義から e，e' は $(E, E') \in r^{\mathcal{E}}$ であるような任意の評価文脈 E，E' の下での振る舞いが等しいことがわかる．したがって，e，e' が値に評価されるような場合については e，e' が同じように振る舞う項であることがわかる．また e，e' が値に評価されない場合は，明らかに任意の E の下で e，e' の振る舞いが等しいことがわかる．

本研究では Belo ら [1] に従って型の二項関係 $*(\theta, \delta)$ を定義する．$*(\theta, \delta)$ の厳密な定義は省略するが，$*(\theta, \delta)$ に含まれる型のうち，一方は well-formed である．詳しくは [15] を参照されたい．

論理関係は開いた項について定義される．

定義 4.4. Γ を型環境，θ，δ を定義 *4.3* で述べたものと同様の写像とする．このとき Γ，θ，δ に対して以下の条件が成り立つとき，$\Gamma \vdash \theta;\delta$ と書く．

1. $\forall \alpha \in \Gamma.\ \alpha \in dom(\theta)$
2. $\forall x{:}T \in \Gamma.\ \exists v, v'.\ x \mapsto v, v' \in \delta$ かつ $(v, v') \in T(\theta, \delta)$

定義 4.5 (論理関係)**.** 論理関係 $\simeq$ は (Γ, e, e', T) と (Γ, T, T') を要素とし，次の条件を満たす集合である．ただし，$(\Gamma, e, e', T) \in \simeq$ を $\Gamma \vdash e \simeq e' : T$，$(\Gamma, T, T') \in \simeq$ を $\Gamma \vdash T \simeq T' : *$ と書く．

$$\Gamma \vdash e \simeq e' : T \quad \textit{iff} \quad \forall \theta, \delta.\ \Gamma \vdash \theta;\delta \text{ ならば } (\theta_1(\delta_1(e)), \theta_2(\delta_2(e'))) \in T(\theta, \delta)$$

$$\Gamma \vdash T \simeq T' : * \quad \textit{iff} \quad \forall \theta, \delta.\ \Gamma \vdash \theta;\delta \text{ ならば } (T, T') \in *(\theta, \delta)$$

論理関係を用いて項が文脈等価であることを証明するためには，論理関係が文脈等価性に対して健全であることを証明する必要がある．

定理 4.1 (健全性)**.**

1. $\Gamma \vdash e \simeq e' : T$ ならば $\Gamma \vdash e \approx e' : T$.
2. $\Gamma \vdash T \simeq T' : *$ ならば $\Gamma \vdash T \approx T' : *$.

証明. 文脈等価性の存在性の証明から，$\simeq$ が adequate で，かつ文脈等価性の置換性，互換性が成り立つことを示せば十分である．証明は以下の手順に従って行う．

1. 型付項や well-formed な型は論理関係によって自身と関係付けられているという仮定のもとで，論理関係が互換性を満たすことを示す．例えば関数適用に関する規則に対しては以下の補題

   $\Gamma \vdash v_{12} \simeq v_{12} : T_1$，$\Gamma, x{:}T_1 \vdash T_2 \simeq T_2 : *$ が成り立つとする．
   このとき $\Gamma \vdash v_{11} \simeq v_{21} : (x{:}T_1 \to T_2)$ かつ $\Gamma \vdash v_{12} \simeq v_{22} : T_1$
   ならば $\Gamma \vdash v_{11}\,v_{12} \simeq v_{21}\,v_{22} : T_2\,[v_{12}/x]$.

   が成り立つことを示す．
2. 型付項や well-formed な型は論理関係によって自身と関係付けられることを示す．これは型付け導出に関する帰納法と 1. より直ちに得られる．
3. 論理関係が adequate で，かつ文脈等価性の置換性，互換性を満たすことを示す．互換性に関しては 1. と 2. よりただちに証明でき，置換性の証明には 2. を用いる．

□

$\boxed{\Gamma \vdash T_1 <: T_2}$

$$\frac{}{\Gamma \vdash B <: B}\ \text{S\_Base} \qquad \frac{}{\Gamma \vdash \alpha <: \alpha}\ \text{S\_TVar} \qquad \frac{\Gamma, \alpha \vdash T_1 <: T_2}{\Gamma \vdash \forall\, \alpha.\ T_1 <: \forall\, \alpha.\ T_2}\ \text{S\_Forall}$$

$$\frac{\Gamma \vdash T_{21} <: T_{11} \quad \Gamma, x{:}T_{21} \vdash \phi\,(T_{12}, x, \langle\ T_{21}\ \Rightarrow\ T_{11}\ \rangle^l\, x) <: T_{22}}{\Gamma \vdash x{:}T_{11} \to T_{12} <: x{:}T_{21} \to T_{22}}\ \text{S\_Fun}$$

$$\frac{\begin{array}{c} T_1' = \mathrm{unref}\,(T_1) \quad T_2' = \mathrm{unref}\,(T_2) \quad \Gamma \vdash T_1' <: T_2' \quad y \text{ is a fresh variable} \\ \Gamma, x{:}T_1' \vdash \mathrm{casts}\,(T_1)\, x\, \mathbf{imp}\, \mathsf{let}\ y = \langle\ T_1'\ \Rightarrow\ T_2'\ \rangle^l\, x\ \mathsf{in}\ \mathrm{casts}\,(T_2)\, y \end{array}}{\Gamma \vdash T_1 <: T_2}\ \text{S\_Refine}$$

$\boxed{\Gamma \vdash e_1\, \mathbf{imp}\, e_2}$

$$\frac{\forall \sigma.\ \Gamma \vdash \sigma\ \wedge\ \sigma(e_1)\downarrow\ \mathrm{implies}\, \sigma(e_2)\downarrow}{\Gamma \vdash e_1\, \mathbf{imp}\, e_2}\ \text{Imp}$$

図 5. 部分型関係の推論規則

5 アップキャスト除去

型 T_1 が型 T_2 の部分型のとき，T_1 の項は T_2 の項として振る舞うことができる．そのためアップキャスト $\langle\ T_1\ \Rightarrow\ T_2\ \rangle^l$ に T_1 の値を適用したとき，実行時型検査は必ず成功する．そこで本研究では $\mathrm{F}^{\mathrm{fix}}_{\mathrm{H}}$ における最適化手法として，アップキャストを恒等関数に置き換えてもプログラム全体の振る舞いが変わらないことを示す．具体的には，アップキャストと恒等関数が論理関係に含まれることを示すことで，それらが文脈等価であることを証明する．

一般に，アップキャストから恒等関数への置き換えを行った後の項は型付け可能とは限らない．よって項 e から，e に登場するアップキャストを恒等関数に置き換えたような項 e' への変換は，e の型チェックが終わり，e を評価する前に行うことを想定している．また e' が型付けできない項であった場合，e' に登場するアップキャストを恒等関数に置き換えた項と e，e' の振る舞いが同じであることは保証されないため，一般にはアップキャストと恒等関数の交換は一度に全て行う必要がある．

部分型関係の判断 $\Gamma \vdash T_1 <: T_2$ は型環境 Γ と二つの型 T_1，T_2 から構成され，図 5 の推論規則から導出される．推論規則は Belo らが使用したもの [1] とほぼ同様である．基本型，型変数の部分型関係は反射性によってのみ成り立つ．多相関数型については (S_Forall) より型変数を加えた型環境の下で部分型関係を導出する必要がある．(S_Fun) は依存関数型間の部分型関係を導出する推論規則で，引数の型に関する部分型関係では通常と同じように反変性が成り立つ．また，戻り値型 T_{12}，T_{22} には自由に出現することのできる変数の型が異なるため，2 章で与えた型上の変数に項を代入するような写像 ϕ を用いて，T_{12} 上の束縛変数を T_{11} へのキャストの適用に置き換えた上で，戻り値型の部分型関係を導出する．(S_Refine) は篩型に対する推論規則である．篩型が付く値はその型の全ての表明を満たしている．つまり，型から項への写像 casts を，2 章で定義した篩型の表明を全て取り除く写像 unref を用いて

$$\mathrm{casts}\,(T) = \mathtt{fun}\ x : \mathrm{unref}\,(T).\ \langle\ \mathrm{unref}\,(T)\ \Rightarrow\ T\ \rangle^l\, x$$

とすると，任意の閉じた篩型 T の閉じた値 v に対して，$\mathrm{casts}\,(T)\, v \longrightarrow^* v$ が成り立つ．篩型 T_1 が篩型 T_2 の部分型であったとき，T_1 の任意の値 v は T_2 の値として振る舞うことができるため，$\mathrm{casts}\,(T_2)\, v \longrightarrow^* v$ が成り立つ．そのため篩型の部分型関係を導くには，$\mathrm{unref}\,(T_1)$ と $\mathrm{unref}\,(T_2)$ の部分型関係と，T_1 の表明が全て満たされるとき必ず T_2 の表明全て満たされることを示さなけれ

ばならない．後者は推論規則 Imp によって導かれる．項を評価するためにはその項が閉じていなければならないが，一般に $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ の型は変数，型変数に対して開かれている．そこで型の表明が満たされることを検査するために，型環境が well-formed であることを維持するような，変数 (または型変数) から閉じた値 (または閉じた型) への写像を 4 章で述べた θ，δ を用いて定義する．

定義 5.1. Γ を型環境とし，σ を型変数から閉じた型，変数から閉じた値への写像とする．このとき Γ と σ に以下の条件が成り立つとき，$\Gamma \vdash \sigma$ と書く．

$$\exists \theta, \delta.\ \Gamma \vdash \theta; \delta \text{ かつ } \sigma = \theta_1 \circ \delta_1$$

以上のような部分型関係を用いて定義されたアップキャストが，論理関係において恒等関数と関係付けられていることを示す．

定理 5.1 (アップキャスト除去)**.** 型環境 Γ，型 T_1，T_2 に対して $\Gamma \vdash T_1 <: T_2$，$\Gamma \vdash T_1$，$\Gamma \vdash T_2$ が成り立つとする．このとき $\Gamma \vdash \langle\ T_1\ \Rightarrow\ T_2\ \rangle^l \simeq (\mathtt{fun}\ x : T_1.\ x) : T_1 \to T_2$ が成り立つ．

証明. これは Belo らと同様に，$\Gamma \vdash T_1 <: T_2$ の導出に関する帰納法で証明できる．(S_Refine) の場合は，T_1 の表明を満たす任意の値に対して T_2 の表明が成り立つことを用いて証明する． □

6 関連研究

$\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ はソフトウェアの契約情報を篩型の表明として静的型情報に埋め込む顕在的契約計算体系 λ^H [6, 8] を多相関数型と再帰関数で拡張した計算体系である．Knowles ら [8] は λ^H の項にアップキャストを挿入してもプログラム全体の振る舞いが変わらないことを証明するため論理関係を用いたが，λ^H では部分型関係による subsumption rule を用いて恒等関数にアップキャストと同じ型を付けることができるため，λ^H の論理関係にある二項はともに同じ型を付けることができた．$\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ や λ^H などにおける高階関数に契約を付与する方法は Findler ら [5] によって考案されたが，Findler らが示した計算体系は契約が静的型情報に現れないものであった．

本研究では，アップキャストが除去可能であることを示すため $\top\top$ 閉包を用いて $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ に対し論理関係を与えた．$\top\top$ 閉包は Pitts によって提案された手法で，Pitts は $\top\top$ 閉包を用いて多相関数型，存在型，再帰関数のある型付ラムダ計算に対し論理関係を与え，それが文脈等価性に対し健全かつ完全であることを示した [11]．その他にも文脈等価性に対し健全かつ完全な論理関係を，Dreyer らが再帰型 [2] や参照型 [3] を含むラムダ計算に対して与えた．文脈等価性を示す手法としては論理関係の他に双模倣などが知られており，Sumii らは文脈等価性に対して，健全かつ完全な双模倣を与えた [14, 13] ．また本研究の論理関係は型付けされた項と必ずしも型付けされない項を含む関係であったが，このようにそれぞれ性質の異なった項を関係付ける論理関係に，Hur らが与えた，ML に似たプログラミング言語と自己書き換えが可能な小さなアセンブリ言語の間の論理関係 [7] がある．Hur らはこの論理関係に対して，全ての部分項が論理関係にあるような項もまた論理関係にある，という基本定理 (*Fundamental Theorem*) が成り立つことを示した．

7 結論

本研究では静的型システムとプログラム実行時の型検査の仕組みをもつ計算体系 F_{H} を再帰関数で拡張した計算体系 $\mathrm{F}_{\mathrm{H}}^{\mathrm{fix}}$ に対し，弱型付文脈等価性とそれに対し健全な論理関係を与えた．さらに，部分型関係が成り立つようなキャストであるアップキャストと恒等関数が論理関係に含まれることを証明することで，アップキャストを除去してもプログラム全体の振る舞いが変わらないことを示した．

謝辞

“refinement type” を訳すにあたって,「篩型」を含め数多くのアイデアを出してくれた末永幸平氏と，有益な助言をして頂いた PPL 2012 の査読者の方々に感謝する．

参考文献

[1] João Filipe Belo, Michael Greenberg, Atsushi Igarashi, and Benjamin C. Pierce. Polymorphic contracts. In *Proc. of ESOP*, LNCS, pages 18–37. Springer-Verlag, 2011.

[2] Derek Dreyer, Amal Ahmed, and Lars Birkedal. Logical step-indexed logical relations. *Logical Methods in Computer Science*, 7(2), 2011.

[3] Derek Dreyer, Georg Neis, and Lars Birkedal. The impact of higher-order state and control effects on local relational reasoning. In *Proc. of ACM ICFP*, pages 143–156, New York, NY, USA, 2010. ACM.

[4] Matthias Felleisen and Robert Hieb. The revised report on the syntactic theories of sequential control and state. *Theor. Comput. Sci.*, 103:235–271, September 1992.

[5] Robert Bruce Findler and Matthias Felleisen. Contracts for higher-order functions. In *Proc. of ACM ICFP*, pages 48–59, New York, NY, USA, 2002. ACM.

[6] Cormac Flanagan. Hybrid type checking. In *Proc. of ACM POPL*, pages 245–256, New York, NY, USA, 2006. ACM.

[7] Chung-Kil Hur and Derek Dreyer. A kripke logical relation between ml and assembly. In *Proc. of ACM POPL*, pages 133–146, New York, NY, USA, 2011. ACM.

[8] Kenneth Knowles and Cormac Flanagan. Hybrid type checking. *ACM Trans. Program. Lang. Syst.*, 32:6:1–6:34, February 2010.

[9] Søren B. Lassen. *Relational Reasoning about Functions and Nondeterminism.* ds, daimi, December 1998.

[10] James H. Morris. *Lambda-Calculus Models of Programming Languages.* PhD thesis, MIT, 1968.

[11] Andrew M. Pitts. Typed operational reasoning. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7, pages 245–289. The MIT Press, 2005.

[12] Gordon D. Plotkin. λ-definability in the full type hierarchy. In J. P. Seldin and J. R. Hindley, editors, *Combinatory Logic, Lambda Calculus and Formalism.* Academic Press, 1980.

[13] Eijiro Sumii. A complete characterization of observational equivalence in polymorphic λ-calculus with general references. In *Proc. of Computer Science Logic (CSL'09)*, pages 455–469, Berlin, Heidelberg, 2009. Springer-Verlag.

[14] Eijiro Sumii and Benjamin C. Pierce. A bisimulation for type abstraction and recursion. *Journal of the ACM*, 54, October 2007.

[15] 関山 太朗 and 五十嵐 淳. 顕在的契約計算におけるアップキャスト除去 (フルバージョン). `http://www.sato.kuis.kyoto-u.ac.jp/~t-sekiym/papers/fh-upcast/full.pdf`.