

題名	論証支援システム EUODHILOS
目的	論証支援システムの研究は、ユーザが与えた（言語系と導出系からなる）論理系を用いて記述した論理的モデルに関する証明等の操作を計算機の支援の下で行うための汎用システムの実現を目指す。
概要及び特徴	<p>論証支援システムEUODHILOSの特徴：</p> <p>(1)「全ての議論領域はそれぞれの論理構造を持つ(Every universe of discourse has its logical structure.)」という認識、哲学を反映する汎用システムである。</p> <p>(2) 使い易い証明構築支援環境を提供する。</p> <p>(3) 人間の論理的思考、記号・論理操作過程を支援する。</p> <p>現在の主な機能：</p> <p>(1) 論理系定義機能 言語系, 導出系(公理、推論・書き換え規則)を定義する機能</p> <p>(2) 証明構築支援機能 思考シート, 木形式の証明, 多様な証明形態</p> <p>(3) 論証のための視覚的インタフェース 論理式エディタ, ソフトウェアキーボード, デスクアクセサリ等</p>
構成	

(1) 論理の定義

まず、確定節文法の記法により言語系を定義する。
 例えば、直観主義的型理論における judgement は次のように定義される：

judgement --> *term, epsilon, type* ;
epsilon --> "ε" ;

次に、推論規則、派生規則、書き換え規則を自然演繹法スタイルの表現法を用いて定義する。例えば、λ導入規則及び否定規則はそれぞれ次のように定義される：

$$\frac{\begin{array}{c} [x \in A] \\ \vdots \\ F[x] \in B \end{array}}{\lambda x. F[x] \in A \supset B} \quad (\lambda\text{-I}) \qquad \frac{A \supset \perp}{\sim A} \quad (\text{def})$$

(2) 定義された論理における証明構成

思考シート上の証明は木形式で対話的に構成される。構成は、前向き、後ろ向きの推論及びいくつかの証明断片の結合、分離を通じて進められる。直観主義的型理論の証明例は次のように進められる。

$$\frac{\begin{array}{c} [x \in P]^1 \\ \hline [f \in (P \vee (P \supset \perp)) \supset \perp]^2 \end{array} \quad \frac{\hline \text{inl}(x) \in P \vee (P \supset \perp)}{(\text{inl-I } (1))} \quad \hline}{\text{f}\bullet\text{inl}(x) \in \perp} \quad (\supset\text{-E } (2, 1))$$

$$\frac{\text{f}\bullet\text{inl}(x) \in \perp}{\lambda x. \text{f}\bullet\text{inl}(x) \in P \supset \perp} \quad (\lambda\text{-I } (2))$$

$$\frac{\lambda x. \text{f}\bullet\text{inl}(x) \in P \supset \perp}{\text{inr}(\lambda x. \text{f}\bullet\text{inl}(x)) \in P \vee (P \supset \perp)} \quad (\text{inr-I } (2))$$

$$\frac{\text{inr}(\lambda x. \text{f}\bullet\text{inl}(x)) \in P \vee (P \supset \perp) \quad [f \in (P \vee (P \supset \perp)) \supset \perp]^2}{\text{f}\bullet\text{inr}(\lambda x. \text{f}\bullet\text{inl}(x)) \in \perp} \quad (\supset\text{-E } (2))$$

$$\frac{\text{f}\bullet\text{inr}(\lambda x. \text{f}\bullet\text{inl}(x)) \in \perp}{\lambda f. \text{f}\bullet\text{inr}(\lambda x. \text{f}\bullet\text{inl}(x)) \in (P \vee (P \supset \perp)) \supset \perp} \quad (\lambda\text{-I } (3))$$

$$\frac{\lambda f. \text{f}\bullet\text{inr}(\lambda x. \text{f}\bullet\text{inl}(x)) \in (P \vee (P \supset \perp)) \supset \perp}{\lambda f. \text{f}\bullet\text{inr}(\lambda x. \text{f}\bullet\text{inl}(x)) \in \sim\sim(P \vee \sim P)} \quad (\text{def } (4))$$

(実際の画面構成例は次頁に示す。)

論証向けインタフェース

LOGIC intu_type

logic_desk_calculator

dyn hoare indu intu modal pred type ** new ** ** end **

valid

SOFT_KEYBOARD

wff_editor

(inr I (2))
 inr ((λx. f0inl (x)) ePV (P2L)) (f0(PV (P2L)) 2L) (DE (2))
 f0inr ((λx. f0inl (x)) eL) (λI (1))
 λf. f0inr ((λx. f0inl (x)) e(PV (P2L)) 2L) (def (1))
 λf. f0inr ((λx. f0inl (x)) e~(PV (P2L)) 2L) (def (1))
 λf. f0inr ((λx. f0inl (x)) e~(PV~P2L)) (def (1))
 λf. f0inr ((λx. f0inl (x)) e~(~(PV~P)) (def (1))

直観主義的型理論と構成的証明

LOGIC intu_type

dynamic hoare induction intu_type modal pred type ** new ** ** end **

FONT
 SOFT_KEYBOARD
 SYNTAX
 INFERENCE_RULE
 REWRITING_RULE
 DERIVED_RULE
 AXIOM
 SHEET_OF_THOUGHT

SYNTAX : intu_type

Judgement --> term1, contain, type;
 term1 --> lambda, variable, ".", term2;
 term1 --> term2;
 term2 --> variable, ops, term3;
 term2 --> meta_var1, ("meta_var, ");
 term2 --> term3;
 term3 --> ("(", term1, ")");
 term3 --> not, term3;
 term3 --> in, ("(", term1, ")");
 term3 --> variable;
 term3 --> constant;
 term3 --> meta_var;
 type --> type, imply, type1;
 type --> type1;

INFERENCE_RULE : intu_type

name : λI
 [D]EA
 ...
 F0)EA
 λx. F0)EA)B

REWRITING_RULE : intu_type

name : def
 ...
 ~A

SHEET_OF_THOUGHT : intu_type

1
 ExpP
 (f0(PV (P2L)) 2L) inl (x) ePV (P2L) (inl I (1)) (DE (2, 1))
 f0inl (x) eL (λI (2))
 λx. f0inl (x) eP2L (inr I (2))
 inr ((λx. f0inl (x)) ePV (P2L)) (f0(PV (P2L)) 2L) (DE (2))
 f0inr ((λx. f0inl (x)) eL) (λI (1))
 λf. f0inr ((λx. f0inl (x)) e(PV (P2L)) 2L) (def (1))
 λf. f0inr ((λx. f0inl (x)) e~(PV (P2L)) 2L) (def (1))
 λf. f0inr ((λx. f0inl (x)) e~(PV~P2L)) (def (1))
 λf. f0inr ((λx. f0inl (x)) e~(~(PV~P)) (def (1))

Hoare論理とプログラムの検証

SYNTAX : hoare

```

n_formula --> formula, left_brace, program, rll
ght_brace, formula;

in
program --> program, semi_colon, program1;
program --> program1;

in
program1 --> assignment_statement;
program1 --> while, formula, "do", program, "od";
program1 --> if, formula, "then", program, "elst";
program1 --> "if";
program1 --> "(", program, ")";

```

INFERENCE_RULE : hoare

```

name : repeat

FAG(A)F

P(whileGdoAod)FA~G

```

hoare

FONT
SOFT_KEYBOARD
SYNTAX
INFERENCE_RULE
REWRITING_RULE
DERIVED_RULE
AXIOM
SHEET_OF_THOUGHT

REWRITING_RULE : hoare

name : arith

```

z=y!
z*(y+1)=(y+1)!

```

AXIOM : hoare

```

TrueQ=0!
z=y!A~(y=x)z=y!
z=y!A~x=z*x!
B(T) (X=T)P(X)

```

DERIVED_RULES

$$\frac{\text{trueQ} \wedge 0! \quad 0!(z=1)z=0!}{\text{true}(z=1)z=0!} \text{(consq1 ())}$$

$$\frac{\text{true}(z=1)z=0! \quad z=0!(y=0)z=y!}{\text{true}(z=1; y=0)z=y!} \text{(comp ())}$$

$$\frac{z=y! \wedge (y=x)z=y! \quad \text{(arith ())} \quad z=y! \wedge x=z*x \wedge (y+1)=(y+1)! \quad z*(y+1)=(y+1)! \quad (y; z=y+1)z=y!}{z=y! \wedge x=y \wedge (y; z=y+1)z=y!} \text{(consq1 ())}$$

$$\frac{z=y! \wedge x=y \wedge (y; z=y+1)z=y! \quad z=y! \wedge x=y \wedge (y; z=y+1)z=y! \quad z=y! \wedge x=y \wedge (y; z=y+1)z=y!}{z=y! \wedge x=y \wedge (y; z=y+1)z=y!} \text{(consq1 ())}$$

$$\frac{z=y! \wedge x=y \wedge (y; z=y+1)z=y! \quad z=y! \wedge x=y \wedge (y; z=y+1)z=y! \quad z=y! \wedge x=y \wedge (y; z=y+1)z=y!}{z=y! \wedge x=y \wedge (y; z=y+1)z=y!} \text{(consq1 ())}$$

$$\frac{z=y! \wedge x=y \wedge (y; z=y+1)z=y! \quad z=y! \wedge x=y \wedge (y; z=y+1)z=y! \quad z=y! \wedge x=y \wedge (y; z=y+1)z=y!}{z=y! \wedge x=y \wedge (y; z=y+1)z=y!} \text{(consq1 ())}$$

Dynamic論理とプログラムに関する論証

**FONT
SOFT_KEYBOARD
SYNTAX
INFERENCE_RULE
REWRITING_RULE
DERIVED_RULE
AXIOM
SHEET_OF_THOUGHT**

SYNTAX : dynamic

```

dynamic_formula --> left_diamond, regular_program, right!
_diamond, formula3;
dynamic_formula --> left_box, regular_program, right_box!
, formula3;

formula --> formula, equivalence, formula0;
formula --> formula0;

formula0 --> formula0, imply, formula1;
formula0 --> formula1;

formula1 --> formula1, or, formula2;
formula1 --> formula2;

formula2 --> formula2, and, formula3;
formula2 --> formula3;

formula3 --> "(", formula, ")";
formula3 --> not, formula3;
formula3 --> dynamic_formula;

```

REWRITING_RULE : dynam

```

name : def

(A)P
~(A)~P

```

AXIOM : dynamic

```

[Q]P (Q)P
(X=Y)P (X)P(T)
(A)P (A) (B)P
~(A)P (A)P (B)P
P (X)A (X)P (T)
x=0 (x=0 true)
((x=0)?>true) (x=0 true)
!x0Axn+1 < (x=0)?> (x=0+1)
x=0+1 < x: x=0 > (x=0+1)
x=0+1 < x: x-1 > (x=0)
!x0x1 = !x0x0 [x: x=0] (x(x-1) !x0)
!x(x-1) = !x1 [x: x-1] (x0x1 !x0)
!x0 [z=1] (x0 [z=1])
!x0x1 = !x1 [x=0?] (z=1)

```

INFERENCE_RULE : dynamic

```

name : invar

P(A)P
P(A)P

```

DERIVED_RULES

$$\frac{!x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0]}{!x0x1 = !x0x0 [x: x=0]} \text{(comp2 ())}$$

$$\frac{!x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0]}{!x0x1 = !x0x0 [x: x=0]} \text{(comp2 ())}$$

$$\frac{!x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0]}{!x0x1 = !x0x0 [x: x=0]} \text{(comp2 ())}$$

$$\frac{!x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0]}{!x0x1 = !x0x0 [x: x=0]} \text{(comp2 ())}$$

$$\frac{!x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0]}{!x0x1 = !x0x0 [x: x=0]} \text{(comp2 ())}$$

$$\frac{!x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0]}{!x0x1 = !x0x0 [x: x=0]} \text{(comp2 ())}$$

$$\frac{!x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0] \quad !x0x1 = !x0x0 [x: x=0]}{!x0x1 = !x0x0 [x: x=0]} \text{(comp2 ())}$$