DiaLog
A system for dialogue logic

Jürgen Ehrensberger[1] and Claus Zinn[2]

[1] Ecole polytechnique fédérale de Lausanne, Laboratoire de Télécommunications
CH-1024 Lausanne, Switzerland
ehrensbe@tcomhp20.epfl.ch
[2] Universität Erlangen-Nürnberg, Lehrstuhl für Künstliche Intelligenz
D-91058 Erlangen-Tennenlohe, Germany
zinn@informatik.uni-erlangen.de

Abstract. We present a proof system which implements dialogue logic as originally developed by Lorenzen. Intuitionistic and classic logic are pre-defined. A rule language allows to easily define dialogue rules for other logics. **DiaLog** provides multi-sorted logic. The system supports fully automated and interactive proof search. A user-friendly graphical interface displays the dialogue tableau, a dialogue trace field and a dialogue strategy grapher.

1 Dialogue logic

The principles of dialogue logic were proposed by Paul Lorenzen in 1958 in order to provide a new autonomous foundation of intuitionistic logic (cf. [5]). In this new approach the validity of a given formula is examined in a formal dialogue between two participants: the *proponent* and the *opponent*. At the beginning of a dialogue the proponent asserts a *thesis* and the opponent may assert *hypotheses*. The consecutive steps of the dialogue are *attacks* upon previous assertions or *defences* against previous attacks which are stated by the proponent and the opponent in turn. Some of these statements may contain new assertions and thus might be subject to subsequent attacks.

1.1 Dialogue rules

A formal dialogue is defined in terms of *particle rules*, also called *argumentation forms*, and *frame rules*. For each logical connective ϕ a particle rule is given which specifies how attacks upon assertions having ϕ as main connective and defences against such attacks have to be performed. Tab. 1 gives an overview of these particle rules.

A logical disjunction asserted by one of the formal dialogue participants may be attacked by its counterpart by stating $'?'$. The first player is then to choose the disjunct that he wants to assert in the defence. In the case of a logical conjunction the attacking player chooses the subformula (either left or right conjunct) that is to be asserted by the other player in the defence. To attack a logical implication,

Assertion	Attack		Defence	
$A \vee B$	$?$		A	B
$A \wedge B$	$L?$	$R?$	A	B
$A \rightarrow B$	$?, A$		B	
$\neg A$	$?, A$			
$\exists x A(x)$	$?$		$A(n)$	
$\forall x A(x)$	$n?$		$A(n)$	

Table 1. Particle rules

the premise of this formula has to be asserted. The defence is carried out by asserting the consequence of the formula. A formula $\neg A$ is attacked by stating A. In this case, there is no possible defence. After an attack to an existential quantified formula $\exists x A(x)$, the defending player can choose a term n for which he asserts the formula $A(n)$. In the case of a universal quantified formula the attacking player chooses the term.

While the particle rules impose restrictions on how to attack propositions and how to defend oneself against such attacks, the frame rules impose restrictions on when attacks and defences may take place in the dialogue. I.e., the frame rules order the exchange of arguments and thus determine the frame in which the dialogue may develop. The frame rules for intuitionistic logic are:

1. Any formal dialogue starts with the proponent's assertion of the thesis. The opponent is free to state an arbitrary number of hypotheses. This defines the setting of the dialogue. The consecutive dialogue steps are performed by the opponent and the proponent in turn. The opponent starts the argument in attacking the thesis.
2. After the setting of the dialogue has been fixed, each action of a player is either an attack upon a previously stated assertion or a defence against an attack of the other player.
3. Each action of the opponent is a reaction (according to the particle rules) upon the *immediately preceding action* of the proponent.
4. A defence of the proponent against an attack of the opponent may only be carried out once.
5. Attacks to which no defence has yet been stated are called *open attacks*. If the proponent[3] wants to perform a defence against an open attack, he has to answer the most recent open attack of the dialogue. If he is not able to answer this attack, he is not allowed to answer any other open attack.
6. It is not allowed to attack atomic formulae. The opponent may state any atomic formula. The proponent may only state an atomic formula after it has been stated by the opponent.
7. A dialogue is won by a player, if the other player cannot perform any action that is conform to the dialogue rules.

[3] Due to the third frame rule there are no open attacks of the opponent.

The frame rules and their proper justification have been the key issue in the development of dialogue logic. The frame rules 1-7 for intuitionistic logic can be modified in order to provide classical logic.

A single dialogue has no significance for the validity of a thesis. Instead, all possible dialogues starting from this thesis have to be considered. The proponent is said to have a *winning strategy* for a formula, if he is able to win all possible dialogues with this formula as thesis by appropriate choices of his statements. The possible dialogues about a formula can be represented as a tree of statements. A winning strategy for a proponent is then a tree with the following properties:

1. All branches of the tree are finite.
2. The leafs of the tree are statements of the proponent[4].
3. No branch can be extended beyond the leaf in accordance with the dialogue rules[5].
4. When it is the proponents turn to perform a statement, only one of the possible alternatives is contained in the tree.
5. When it is the opponents turn to perform a statement all possible alternatives are contained in the tree.

The winning strategy is not unique.

It can be shown that the notion of winning strategies (wrt. a well-defined set of dialogue rules) coincides with the notion of provability in Gentzen's calculus LJ for intuitionistic logic (cf.[2], [3] for details of this equivalence theorem).

1.2 Finite dialogue trees

In most cases, a dialogue tree (containing all possible dialogues in accordance with the dialogue rules) is infinite. It may contain (a) dialogues with infinitely many steps – due to repeated attacks or repeated defenses – and (b) infinite ramifications – when terms have to be chosen. In order to only allow finite dialogue trees, two measures have to be taken.

Firstly, the number of repeated attacks upon the same formula and the number of repeated defences to the same attack have to be restricted using an *attack limit* and a *defence limit*. It has to be pointed out, that these limits restrict the possibility to find a winning strategy, i.e., with too low limits a strategy for a formula might not be found even if the formula is valid in some given logic.

Secondly, whenever attacks upon universal quantified formulae and defences against attacks of existential quantified formulae are performed the dialogue participants have to choose appropriate terms. These terms are elements of the Herbrand universe which is, in general, infinite. To avoid infinite ramifications,

[4] In DiaLog, the leafs of the tree are resign statements (symbolized by '%') of the opponent.

[5] I.e., there exists no action (wrt. the rules) for the opponent to continue the dialogue.

two purely technical methods are applied that do not affect the possibility to find a strategy.

When it is the opponent that has to choose a term, he introduces a new symbol that has not yet been used in the dialogue. This constitutes the most difficult case for the proponent. If the proponent can find a winning strategy for this case, he will succeed for any other case too.
If the proponent has to choose a term, he delays his choice until only a finite number of possible terms remain. Like the opponent he introduces a new symbol that, in this case, stands for a not yet specified term. When the proponent wants to state an atomic formula containing such a symbol, the finite number of atomic formulae that have already been stated by the opponent is examined. Only the atomic formulae, that can be unified with the proponent's atomic formula are considered. The unification results in concrete values for the formerly unspecified term. These symbols that are introduced into the dialogue by the opponent or the proponent are called *term symbols* and are prefixed in **DIALOG** with '$'.

1.3 An example

Those who do not know dialogue logic may appreciate an example. Fig. 1 shows the beginning of the dialogue for the thesis $\neg\neg a \to a$ and the hypothesis $a \vee \neg a$.
After the attack of the thesis by the opponent the proponent cannot state the

Opponent		**Proponent**
$a \vee \neg a$	1	$\neg\neg a \to a$
$?, \neg\neg a$	2	

Fig. 1. Starting of a dialogue about $\neg\neg a \to a$ under TND[6]

atomic formula a. He has the choice to attack either the hypothesis $a \vee \neg a$ or the assertion $\neg\neg a$. When he chooses to attack $\neg\neg a$ he might lose the dialogue. This situation is shown in Fig. 2 and Fig. 3. In line 3 of both dialogues the proponent

Opponent		**Proponent**
$a \vee \neg a$	1	$\neg\neg a \to a$
$?, \neg\neg a$	2	$?, \neg a$
$?, a$	3	$?1$
a	4	%

Fig. 2. Dialogue continuation 1

Opponent		**Proponent**
$a \vee \neg a$	1	$\neg\neg a \to a$
$?, \neg\neg a$	2	$?, \neg a$
$?, a$	3	$?1$
$\neg a$	4	$?, a$
%	5	

Fig. 3. Dialogue continuation 2

[6] abbrev. for Tertium Non Datur, law of excluded middle.

cannot defend the attack of the opponent as there is no defence against an attack upon a negation. Moreover, he cannot use the atomic formula a of the opponent as defence against the attack of the opponent in line 2, as this is not the most recently stated attack[7]. The only remaining possibility is to attack the hypothesis. In dialogue continuation 1 the opponent takes the right choice and again states the atomic formula a so that the proponent has to resign. However, the proponent wins the dialogue if the opponent chooses the wrong alternative and states the formula $\neg a$ (line 4) as shown in dialogue continuation 2.

If the proponent already attacks the hypothesis in line 2, he can win both dialogues that result from the remaining choices of the opponent. These dialogues are depicted in Fig. 4 and Fig. 5. In this example, the proponent wins the last

Opponent		**Proponent**
$a \vee \neg a$	1	$\neg\neg a \rightarrow a$
$?, \neg\neg a$	2	$?1$
a	3	$a[[2]]$
%	4	

Fig. 4. Dialogue continuation 3[8]

Opponent		**Proponent**
$a \vee \neg a$	1	$\neg\neg a \rightarrow a$
$?, \neg\neg a$	2	$?1$
$\neg a$	3	$?2, \neg a$
$?, a$	4	$?3, a$
%	5	

Fig. 5. Dialogue continuation 4

two dialogues, independent of the actions taken by the opponent. In the first two dialogues the winner depends on the choice of the opponent.

For showing the validity of a proposition P the proponent needs to have a winning strategy, i.e. he must know how to force the opponent to lose. We must therefore look at all possible dialogues for a given thesis P. It is useful to represent these dialogues by dialogue trees (cf. Fig. 6). The dialog tree branches whenever proponent or opponent has a choice. The leaves of the dialogue tree mark the end of dialogues. Note that even for valid formulae some dialogues may be won by the opponent (if the proponent does not apply his winning strategy).

2 DiaLog

DiaLog provides two modes to generate a winning strategy: a fully automatic mode and an interactive mode. In the latter, the user plays the role of the proponent and is thus obliged to choose from the set of alternatives the appropriate statements in order to win the dialogues. The system assures that all possible alternatives of the opponent are considered.

[7] *Note:* Imagine, that the proponent sets the negation $\neg A$ and that the opponent attacks $\neg A$ by setting $?, A$. According to frame rule 5 the proponent is obliged to always defend himself by answering the most recent open attack. But there exists no defence upon attacks on $\neg A$. Thus the proponent cannot defend himself against the most recent open attack and is thus not allowed to answer any other open attacks.

[8] Note that 'a [[2]]' means: 'a is my defence against your attack in line 2'.

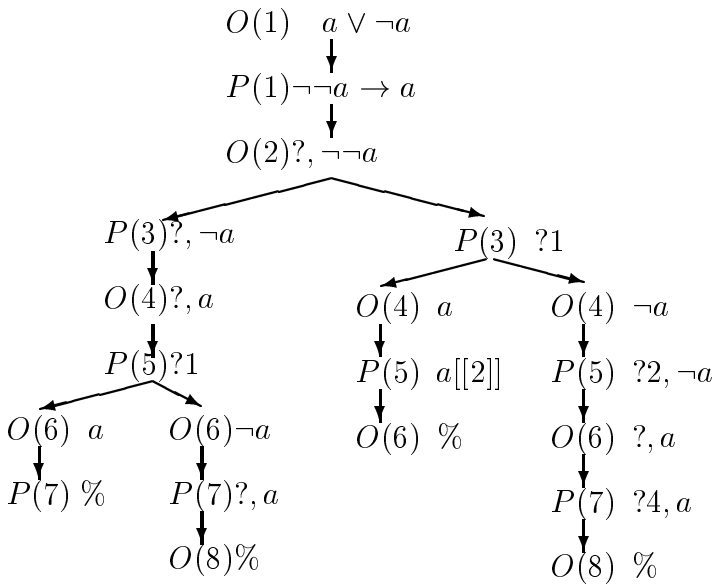

Fig. 6. Dialogue tree for $\neg\neg a \to a$ under TND

For automated mode we give a very simple example, the proof for the Peirce formula, to illustrate the work with **DIALOG**.

```
; *** The PEIRCE FORMULA
declare predicate a () prefix a
declare predicate b () prefix b

hypothesis TND a OR NOT a
thesis a SUB b SUB a SUB a
```

All symbols like constant, variable, and function symbols have to be declared before their use in the specification of thesis and hypotheses. The declarations are followed by one thesis and an arbitrary number of (named) hypotheses.
Fig. 7 shows the main window of **DIALOG** after proof termination of the Peirce formula. The proof has been found automatically. It can be visualized by the strategy grapher (cf. Fig. 8).

We sketch now how to prove the irrationality of $\sqrt{2}$. The formalization (the thesis and two hypotheses) is as follows:

$$\neg\exists p \exists q (2 = \frac{p^2}{q^2} \wedge \neg(\exists s: \ 2s = p \wedge \exists t: \ 2t = q)).$$

$$\forall u (\exists t: \ 2t = u^2 \to \exists s: \ 2s = u)$$
$$\forall x \forall y \forall z (2x = y^2 \wedge 2z = y \to 2z^2 = x)$$

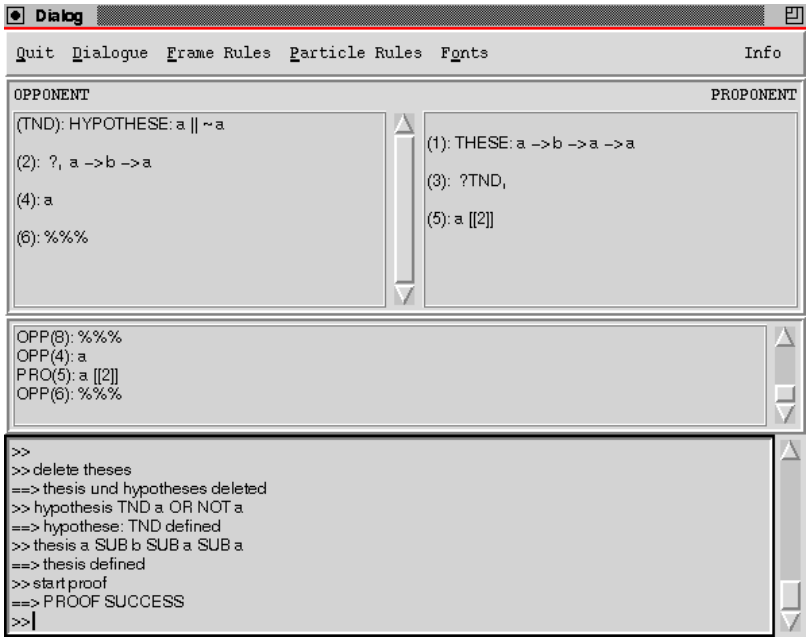

Fig. 7. Main window (screenshot after the Peirce formula has been proven)

In DIALOG this is written in the following form:

```
; *** Irrationality of sqrt 2
declare sort nat
declare constant 2 nat

declare variable p nat \ declare variable q nat \ declare variable r nat
declare variable s nat \ declare variable t nat \ declare variable u nat
declare variable x nat \ declare variable y nat \ declare variable z nat

declare function * (nat nat) -> nat infix 3 4 *
declare function ^ (nat nat) -> nat infix 6 5 ^
declare predicate = (nat nat) infix =

hypothesis A1 ALL u ((EXI t 2 * t = u ^ 2) SUB (EXI s 2 * s = u))
hypothesis A2 ALL x ALL y ALL z ((2 * x = y ^ 2 AND 2 * z = y) SUB 2 * z ^ 2 = x)

thesis NOT EXI p EXI q (2 * q ^ 2 = p ^ 2 AND NOT ((EXI s 2 * s = p) AND (EXI s 2 * s = q)))
set attack limit 2
```

As one can see from the example DIALOG provides multi-sorted logic. Sorts and a hierarchy of sorts can be defined.

The winning strategy for the irrationality of $\sqrt{2}$ has been established in interactive mode. It's structure is shown in Fig. 9.

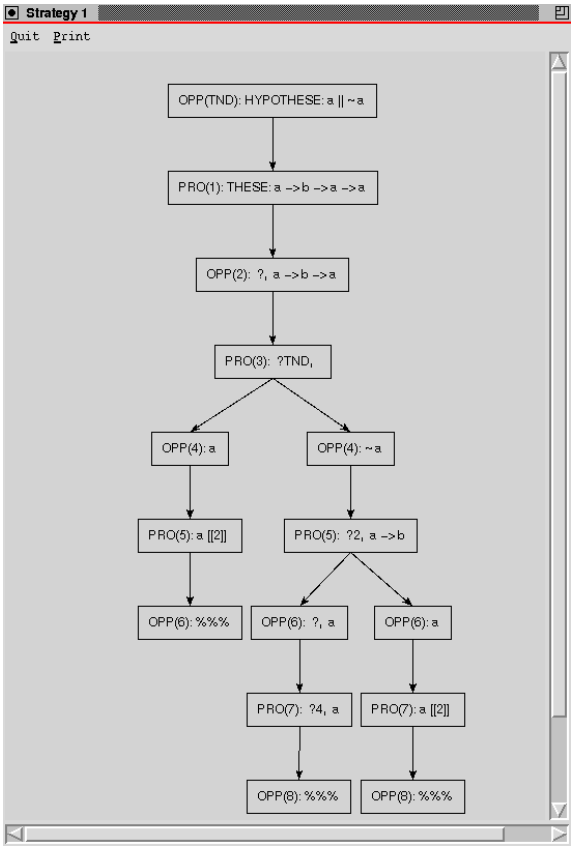

Fig. 8. Winning strategy for the Peirce formula

3 A rule language to redefine dialogue rules

The particle and frame rules that are used for the dialogues are not fixed in DiaLog. There are predefined rules for intuitionistic and classical logic, but the system also provides means to modify or completely redefine these rules. The rule language is embedded in the Scheme programming language that has been used to implement DiaLog. Various Scheme functions are provided which allow to obtain information about the current dialogue, e.g., whether an attack is still open, the position of the most recently stated open attack, the number of attacks already performed upon an assertion.

3.1 Speech acts

We describe the internal format of dialogue speech acts as used in DiaLog. The elements that describe a speech act form a 6-tuple (cf. Tab. 2). The σ contains

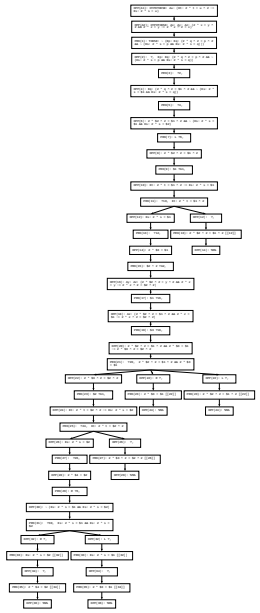

Fig. 9. Proof structure for $\sqrt{2}$ is irrational

element	meaning	value
σ	sign	opponent or proponent
ν	identification	name or number
δ	assertion	formula
η_1	relation	name or number
η_2	type	attack, defence, ...
ζ	parameter	left or right or term

Table 2. A speech act is a 6-tuple

the information which of the players has stated the speech act. The identification ν of a speech act is unique within a dialogue but not within a dialogue tree. It is used to refer to a speech act in subsequent attacks or defences. When referring to a hypothesis, ν contains the name of the hypothesis that the user has attached to it in its declaration. For all other speech acts, ν contains a natural number indicating the dialogue step in which this speech act has been stated. If the speech act contains a formula, δ will contain that formula. For assertion-free statements, δ contains the empty formula. The η_1 element specifies the identification of a previous speech act that is now attacked or defended. The type of the speech act, i.e. whether it is an attack, a defence, a thesis or a hypothesis, is contained

in η_2. The ζ part is required for speech acts that need additional parameters in accordance with the particle rules. In attacks upon conjunctions this element specifies whether the left or the right subformula has to be stated in the defence. For attacks or defences of quantified formulae ζ contains a term.

The terms as well as the formulae in δ are objects of the programming language Scheme that provide numerous methods for testing, modification and selection of subformulae.

3.2 Particle Rules

The application of particle rules create speech acts, which can be set within a dialogue. They are applied to the last step performed in the dialogue. As an example consider a conjunction $A \wedge B$ that has been stated by the proponent in the last dialogue step with the identification 5 as a defence against an attack in step 4. This speech act is described by the 6-tuple

```
(proponent 5 A ∧ B 4 defence empty-parameter).
```

In this case, the particle rules generate two new speech acts that are attacks upon the conjunction[9]:

```
(opponent x empty-formula 5 attack left)
(opponent x empty-formula 5 attack right).
```

The general form to define a particle rule is:

```
(define particle-rule name body).
```

The `name` should be of type string, the `body` is a Scheme function call with five arguments and has one of the following forms:

```
(p-rule ny delta eta1 eta2 zeta
   (conditions condition_1 ... condition_n)
   (speechact-form delta' eta1' eta2' zeta'))

(p-rule ny delta eta1 eta2 zeta
   (conditions condition_1 ... condition_n)
   (speechact-list
     (speechact-form delta_1 eta1_1 eta2_1 zeta_1)
     ⋮
     (speechact-form delta_n eta1_n eta2_n zeta_n)))
```

In the formal parameters `ny`, `delta`, `eta1`, `eta2` and `zeta` are the necessary elements of the speech act under consideration that are provided to the function. When the conditions of the function evaluate to true, one or more speech act forms are generated. *Speech act forms* are 4-tuples, omitting the sign and the identification of regular speech acts. These elements are added when the speech act form is actually set in the dialogue.

A particle rule that generates the speech act forms of the example given above can be defined as:

[9] In the dialogue tableau, these speech acts will appear as 'L?5' and 'R?5', respectively.

```
(define particle-rule
  "A_Con"
  (p-rule ny delta eta1 eta2 zeta
         (conditions           (equal? (main-symbol delta) "AND"))
         (speechact-list
            (speechact-form     empty-formula                  ;; delta
                                ny                             ;; eta1
                                attack                         ;; eta2
                                left )                         ;; zeta
            (speechact-form     empty-formula                  ;; delta
                                ny                             ;; eta1
                                attack                         ;; eta2
                                right))))                      ;; zeta
```

When this rule is applied to a speech act it tests whether a conjunction was stated and in this case generates the required speech act forms.

The rules for quantified formulae have to take into account that a term symbol has to be used instead of a concrete term. The term symbol is chosen by the system when the given speech act is set in the dialogue, not when the speech act is generated, as a speech act might be set multiple times in a dialogue but the term symbol has to be unique. To indicate that a term symbol has to be chosen, the quantor variable is placed into the element ζ of the speech act form instead of a term or a term symbol. This method is used in the following rule that generates attacks upon universal quantified formulae:

```
(define particle-rule
  "A_All"
  (p-rule ny delta eta1 eta2 zeta
         (conditions   (equal? (main-symbol delta)
                               "ALL"))
         (speechact-form     empty-formula                     ;; delta
                             ny                                ;; eta1
                             attack                            ;; eta2
                             (quantor-variable delta))))       ;; zeta
```

The rule that generates defences against such attacks is defined as:

```
(define particle-rule
  "V_All"
  (p-rule ny delta eta1 eta2 zeta
         (conditions   (equal? (main-symbol (delta-function eta1))
                               "ALL")
                       (eqv? eta2 attack))
         (speechact-form    (substitute-variable               ;; delta
                                 (body (delta-function eta1))
                                 (quantor-variable
                                     (delta-function eta1))
                                 zeta)
```

```
                                    ny                                  ;; eta1
                                    defence                             ;; eta2
                                    empty-parameter)))                  ;; zeta
```

The expression `(delta-function eta1)` returns as a result the formula of the speech act who's identification is equal to `eta1`. When this rule is applied to an attack upon a universal quantified formula it takes as assertion of the newly generated speech act the body of the quantor formula and replaces the quantor variable with the term symbol that the system has placed into the element ζ of the attack when the attack was stated in the dialogue.

The same method can be used for the particle rules that treat existential quantified formulae.

3.3 Frame Rules

The frame rules serve to restrict the range of possible speech acts created by the particle rules.

Before speech acts are actually set in the dialogue, the frame rules are applied to the speech acts that have been generated in all previous steps. They determine which of these speech acts are allowed in the current dialogue state. In addition, they delete speech acts that won't be allowed in any subsequent dialogue step.

Frame rules are defined in a form similar to particle rules:

`(define frame-rule` *name body*`).`

The body of the frame rule is slightly different from the body of a particle rule:

`(r-rule sigma delta eta1 eta2 zeta`
`(conditions` $condition_1$ ... $condition_n$`)`
flag`)`

Again, conditions are specified that evaluate the elements of a speech act. The flag can have the values `lock-speechact` and `delete-speechact`, determining the action that has to be taken if all conditions evaluate to true.

An example of a locking rule is the frame rule that locks all speech acts except for the thesis, when the thesis has not yet been set in the dialogue:

```
(define frame-rule
  "RR_D_Thesis_P"
  (r-rule sigma delta eta1 eta2 zeta
          (conditions (not (eqv? eta2 these))
                      (not (thesis-set?)))
          lock-speechact))
```

When the number of attacks of the proponent upon the same dialogue step exceeds the limit of attacks that has been specified, every new attack upon this dialogue step can be deleted. This is performed by the following rule:

```
(define frame-rule
  "RR_D_Angriff_P"
```

```
(r-rule sigma delta eta1 eta2 zeta
         (conditions (eqv? sigma p_signum)
                     (eqv? eta2 attack)
                     (>= (attack-repetitions eta1)
                         attack-limit))
         delete-speechact))
```

The built-in functions `thesis-set?` and `attack-repetitions` provide information about the current dialogue state. There are several other functions that can be used in the frame rules. As different dialogue rules might require different information about the current dialogue a means is provided to construct new information functions.

Using the rule language that has been presented in the examples, rules for ontic and deontic modal logic have been defined in extension to the rules for classical an intuitionistic logic.

3.4 Built-in heuristics for proponent and opponent

At any time of the dialogue the proponent and the opponent have, in general, various choices to continue a given dialogue. The dialogue players have to choose one suitable action from a set of allowed actions (generated by the application of particle and frame rules).
In **DiaLog** we have implemented simple heuristics which determine an action which best serves the goal to minimize the length of dialogues (i.e., the number of nodes to be examined in order to find the proof).

The opponent's heuristics is to prefer setting assertion-free speech acts. If this is not possible he will set speech acts containing complex formulae. He tries to avoid setting atomic formulae.

The proponent's heuristics try to answer open attacks as soon as possible. If the proponent is able to defend himself by setting an atomic formula he wins the dialogue. He will thus prefer to defend himself with atomic formulae rather than with complex formulae. Furthermore, the proponent prefers assertion-free attacks rather than attacks which consist in setting atomic formulae. If not otherwise possible, the proponent will attack with complex formulae. In general, repetitions of speech acts have a low priority.

Please note, that in unsuccessful dialogues all speech acts of the proponent will be tested. The heuristics thus do not minimize the number of nodes to be explored in such dialogues. In the same way, the opponent's heuristics do not minimize the number of nodes in successful dialogues, but only in dialogues which are lost.

4 Earlier and future work

4.1 Earlier work

In the history of dialogue logic there has always been a struggle about the 'correct' specification of particle and frame rules. In the literature on dialogue logic

numerous versions of them (mostly kept rather informal) can be found (e.g., cf. [5]). FELSCHER gives a formal account on dialogue logic including an equivalence theorem which states that provability by winning strategies (with a well-defined set of dialogue rules) coincides with provability in Gentzen's calculus LJ for intuitionistic logic (cf. [2], [3]).

Around 1970 at least 2 implementations of dialogue logic existed. One of them has been made at Erlangen University by HAAS (cf. [4]). Another implementation has been used in Austin, Texas (implemented in Fortran on a Control Data 6600), where LORENZEN spent some time as a visiting professor (around 1970). Due to memory restrictions these systems implemented only propositional logic.

4.2 Future work

As far as the authors know, **DIALOG** is the first system which implements dialogue logic for full predicate logic. We had the following goals in mind:

1. To promote dialogue logic which is rather unknown in the automatic deduction community.
2. To better understand the dialogue calculus.
3. To have a system for teaching purposes.

There are several matters which require further study. Primarily, we are interested in how to find *fast* a winning strategy, that is finding the shortest and most elegant proof. There are several ways to achieve this:

- Avoid superflous dialogue steps by adapting the frame rules such that the equivalence of winning strategies to the notion of provability in intuitionistic logic (classic logic, respectively) is preserved. Therefore, we have provided a mechanism for redefining frame rules easily. It has been our mayor issue to technically support the experimentation with dialogue rules.
- Avoid superflous dialogue steps by modifying the heuristics which sort possible speech actions by their relevance.

To better analyse the system's performance future work will require an in-depth evaluation of statistical information about the proof process including the number of success and failure nodes, the number of hypotheses used, the maximum of attack and defence repetitions, the maximal length of dialogues, the number of dialogues generated etc.

The current version of **DIALOG** includes a strategy grapher which graphically represents the winning strategy if the proof succeeds. It should be noted, however, that large proofs require to improve the strategy grapher (e.g., to provide a zoom facility and the possibility to cut-off branches).

For system evaluation we have written a converter which translates `TPTP` problems into the input format of **DIALOG**. First tests suggest that without

human guidance (thus without running DIALOG in interactive mode) the prover will be unable to find proofs of complex theorems. Currently, we regard the formulae presented to prove the irrationality of $\sqrt{2}$ as a complex theory.

5 Availability

The system has been written and developed in the programming language Scheme. The Scheme implementation supports a purely textual user interface. For the graphical user interface you will require `STk`, a Scheme dialect which interfaces with the X Windows system. DIALOG has been developed within the master's project of the first author. DIALOG contains all the features (and some more) described here. It should still count as a prototypical version. For efficiency reasons some code rewriting may be necessary. DIALOG is freely available and can be obtained by contacting the authors.

Comments and bug reports are highly appreciated.

6 Conclusion

We presented a proof system which implements dialogue logic and which offers the possibility to freely and easily redefine the dialogue rules. Intuitionistic and classic logic are predefined. Multi-sorted logic is supported. DIALOG runs in automated and interactive mode. A user-friendly textual and graphical interface is provided. DIALOG can be used to better understand dialogue logic and its limits.

Acknowledgements: We would like to thank the anonymous referees for their comments.

References

1. Ehrensberger, Jürgen. "Ein System für Dialogische Logik". *Diplomarbeit am Institut füer Mathematische Maschinen und Datenverarbeitung, Lehrstuhl für Künstliche Intelligenz,* Universität Erlangen-Nürnberg. 1996.
2. Felscher, Walter. "Dialogues, Strategies and Intuitionistic Provability". *Annals of Pure and Applied Logic* 28 (1985): 217–254
3. Felscher, Walter. "Dialogues as a Foundation for Intuitionistic Logic". *Handbook of Philosophical Logic.* Vol. III. Ed. by Dov M. Gabbay und Franz Guenthner. Dordrecht: D. Reidel, 1986. 341–372.
4. Haas, Gerrit. "Programme zur dialogischen Logik". *Arbeitsberichte des IMMD. Bd. 3, Nr. 4. Universität Erlangen-Nürnberg, Oktober 1970.*
5. Lorenzen, Paul and Lorenz, Kuno. "Dialogische Logik". *Wissenschaftl. Buchgesellschaft, Darmstadt.* 1978.

This article was processed using the LaTeX macro package with LLNCS style