Interactive System for Arranging Issues based on PROLEG

Ken Satoh,
National Insitute of Informatics
Kazuko Takahashi, Tatsuki Kawasaki,
KwanseiGakuin University

- Background
- PROLEG
- Extension of PROLEG to Arrange Issues
- Demonstration
- Conclusion

Background: Computational Support to Arrange Issues in Civil Litigation

In Japan, we have the procedure of “arranging issues” in civil litigation where we clarify which facts are in dispute and what kind of evidence action should be made.
In the beginning of 2020, the Japanese court decided to use Microsoft Teams for arranging issues and they said that they would start the procedure from May 2020 but due to COVID-19, the procedure seems to be suspended currently.
However, the usage of Microsoft Teams is just to replace face-to-face procedure by the one in distributed environment
⇒ **They apply IT technology to the procedure in a very superficial way and does not make a full use of IT technology.**

Background: PROLEG

- We have been developed "PROlog-based LEGal reasoning support system" (PROLEG) since 2009 to simulate reasoning by judges in litigation.
- Support for Judgement Reasoning in Civil Litigation (2500 rules for civil code and supreme court case law).
- Function of PROLEG: Simulating judgement after finishing all the presentation of both sides and fact finding.

→**In this presentation, we extend PROLEG into an interactive system for arranging issues.**

Example of Claims in Civil Litigation

Alice (plaintiff) claims:
I run door-to-door sales of water purifiers, and on January 15, 2020, when I visited Bob's home and explained about a water purifier, Bob said "It looks good for my health, so I wanted to buy it". So I made a contract with Bob (called "Contract 1") and delivered the water purifier to Bob's home. However, I was in trouble because he didn't pay the price, so I decided to make a litigation. Bob has claimed that the water purifier was a contract made by my menace and was invalid by cancellation, but there is no such fact, and Bob has requested that the water purifier be picked up because he does not like it for some reason. Since Bob said that he would buy it, I don't think Bob can cancel the contract anymore.

Example of Claims in Civil Litigation(continued))

Bob (defendant) claims:
Alice visited my house on January 15, 2020 and asked me to buy a water purifier. I wasn't interested in the water purifier, so I said I didn't need it. Her attitude changed suddenly, and she shouted, ”If you don't buy it, I'll visit you every day until you buy it.” I was scared and said, ”I will buy a water purifier.” After the water purifier was delivered, I feel that this is unreasonable and said to Alice, ”I will cancel the contract for the water purifier, so I want you to pick up the water purifier.” However, Alice said, ”I have already delivered it and will not accept returns. Please pay the price otherwise I will sue you.” and she sued me and I am very embarrassed.

Example of Claims in Civil Litigation(continued))

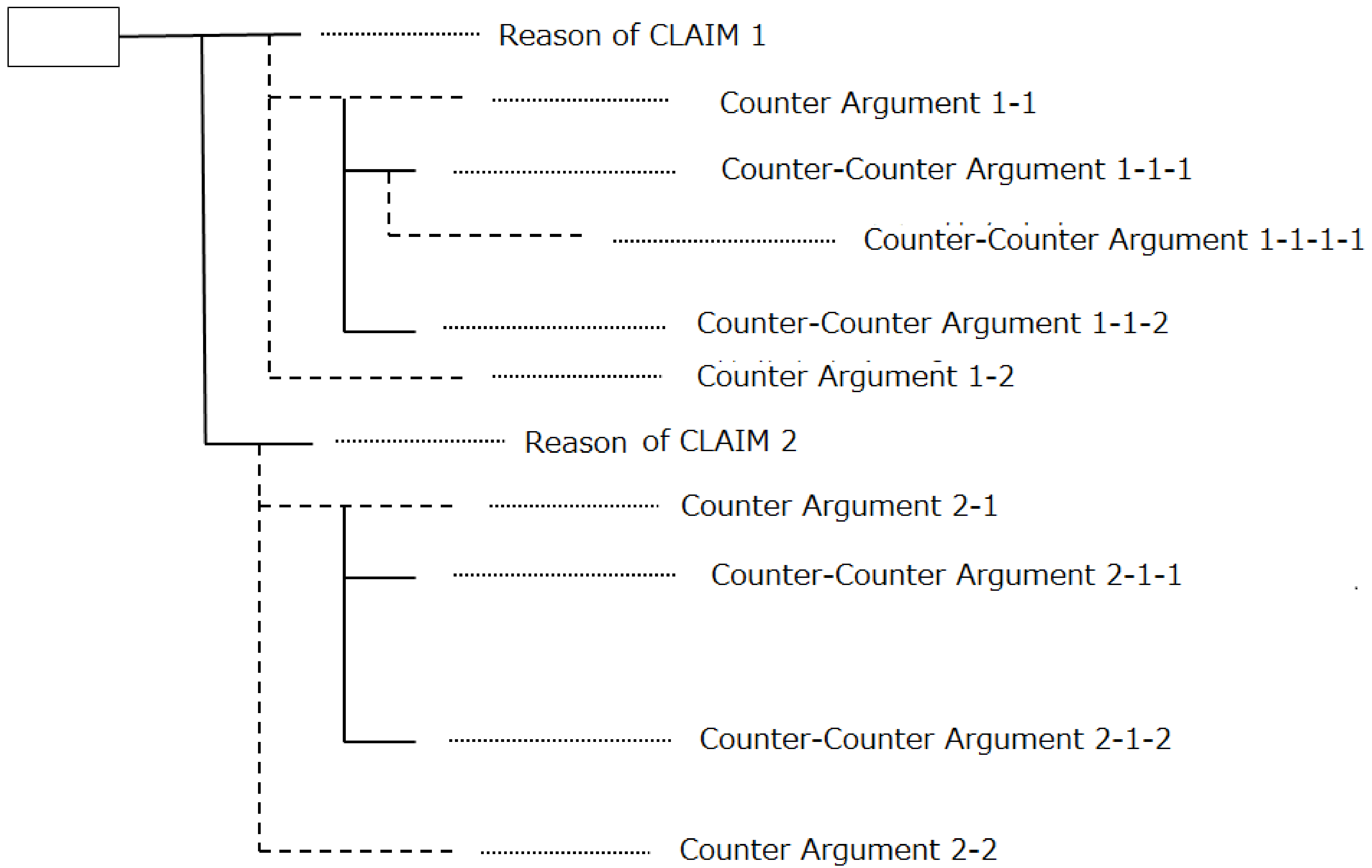
It is necessary to extract the legal facts from the above claims to form an appropriate argument in the court.
→ If any side does not make a claim in a legally correct manner the side will lose the case in the trial.
→ However, the above claims contain various facts, and it is not clear which of them should be selected.
→ PROLEG can choose requirement facts automatically by reasoning about PROLEG rules.

PROLEG

- A PROLEG system consists of a rulebase and a factbase
- PROLEG rulebase consists of the following expression.
  - A **general rule** of the form of Horn clauses (without negation as failure):
    $H \Leftarrow B_1, ..., B_n.$
    Meaning: if $B_1, ..., B_n$ are true H is true in general.
  - An **exception** is an expression of the form $exception(H, E)$ where H, E are atoms each of which is the head of rule.
    Meaning: even if there is a rule which satisfies the condition of rule for H, if E is true H is false.
- PROLEG factbase consists of the truth value of basic facts of the form:
  fact(P).

We can visualize judgement reasoning by a PROLEG block diagram.

PROLEG Block Diagram


Reason of CLAIM 1
Counter Argument 1-1
Counter-Counter Argument 1-1-1
Counter-Counter Argument 1-1-1-1
Counter-Counter Argument 1-1-2
Counter Argument 1-2
Reason of CLAIM 2
Counter Argument 2-1
Counter-Counter Argument 2-1-1
Counter-Counter Argument 2-1-2
Counter Argument 2-2

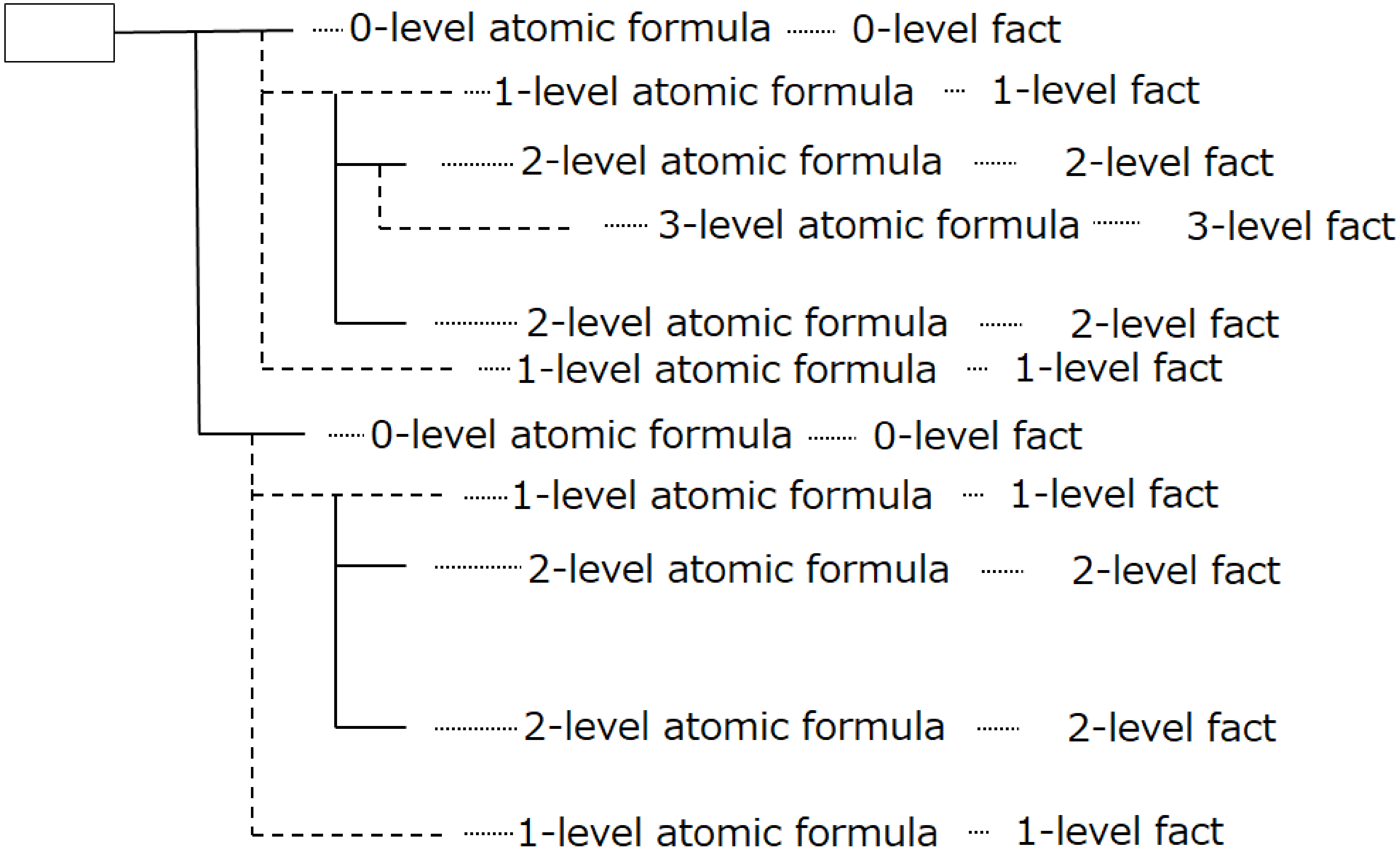
Extension of PROLEG to Arrange Issues: Indexing a level for PROLEG literals

1. First, we define the dependency on the atomic formula that appears in the general rule. First, among the conclusions of the general rule, the conclusion that does not appear in the body of the rule of any general or in the exception of any exception rule is called 0-*level conclusion*. Then, when making a top-down proof tree from the 0-level conclusion using only the general rules, we end up the fact predicates. We call the fact predicates finally visited 0-*level facts* and the 0-level conclusion and the intermediate visited atomic formulas called 0-*level atomic formulas*.

Extension of PROLEG to Arrange Issues: Indexing a level for PROLEG literals(continued))

2. Suppose that the i-level atomic formulas and the i-level facts are decided. For exception rules that conclude with the i-level atomic formula, the collection of the atomic formulas of the exceptions of the such exception rules are called $i+1$-*level exception*. When making a top-down proof tree from the $i+1$-level exception using only the general rules, we end up the atomic formulas. We call the fact predicates finally visited $i+1$-*level facts* and the $i+1$-level exception and the intermediate visited atomic formulas called $i+1$-*level atomic formulas*.

Indexing Atomic Fomulas


0-level atomic formula 0-level fact
1-level atomic formula 1-level fact
2-level atomic formula 2-level fact
3-level atomic formula 3-level fact
2-level atomic formula 2-level fact
1-level atomic formula 1-level fact
0-level atomic formula 0-level fact
1-level atomic formula 1-level fact
2-level atomic formula 2-level fact
2-level atomic formula 2-level fact
1-level atomic formula 1-level fact

Interaction Process in PROLEG

1. Let the plaintiff choose one of the following 0-level conclusion. This is the plaintiff's claim.
2. We make a top-down proof tree from 0-level conclusion and when we encounter the 0-level fact, we ask the plaintiff if the plaintiff claims that fact. If the plaintiff claims it, then we let the plaintiff to instantiate variables in 0-level facts and added instantiated facts to the fact base and we reflect the relevant part of the proof tree with such instantiation. If the plaintiff does not claim it, we delete the path related with the fact in a proof tree.
3. If all the concrete facts are entered, the modified proof tree is displayed.

Interaction Process in PROLEG(continued))

4. Suppose that a user finished to enter $turn$-level facts. We make a proof tree for $turn + 1$-level exception and when we encounter the $turn + 1$-level fact,

    - if $turn + 1$ is odd, we ask the defendant if the defendant claims that fact. If the defendant claims it, then we let the defendant to instantiate variables in $turn + 1$-level facts and added instantiated facts to the fact base and we reflect the relevant part of the proof tree with such instantiation. If the defendant does not claim it, we delete the path related with the fact in a proof tree. We also ask the defendant whether the defendant admits $turn$-level fact or not, we make a label of the fact as “issue to be determined”.

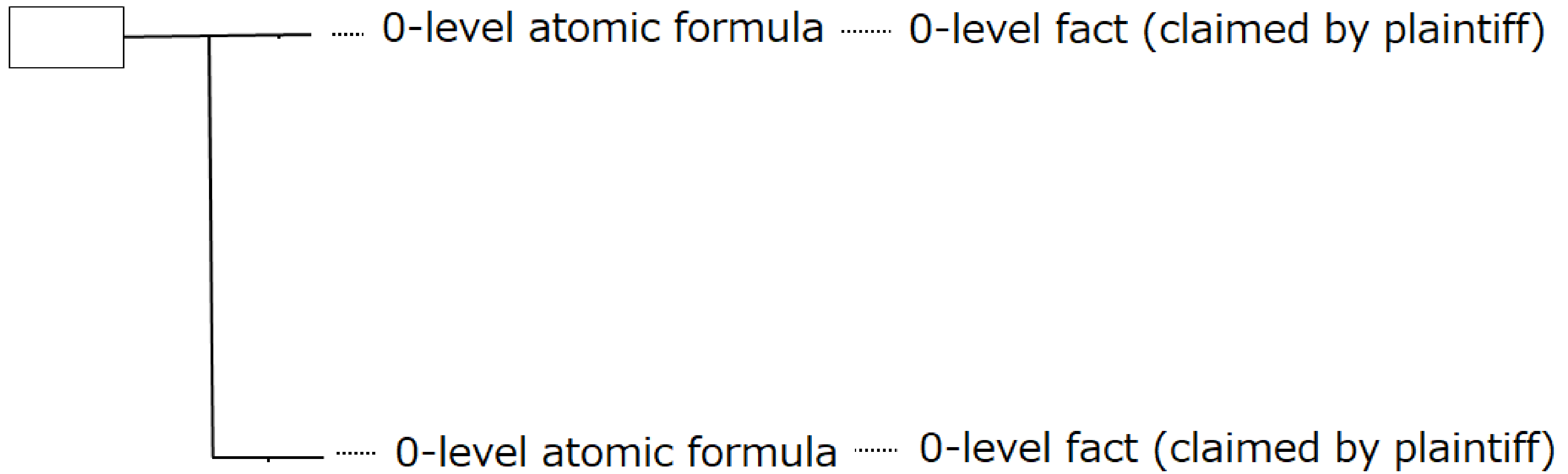
Interaction Process in PROLEG(continued))

- if $turn + 1$ is even, we ask the plaintiff if the defendant claims that fact. If the plaintiff claims it, then we let the plaintiff to instantiate variables in $turn + 1$-level facts and added instantiated facts to the fact base and we reflect the relevant part of the proof tree with such instantiation. If the plaintiff does not claim it, we delete the path related with the fact in a proof tree. We also ask the plaintiff whether the plaintiff admits $turn$-level fact or not, we make a label of the fact as "issue to be determined".

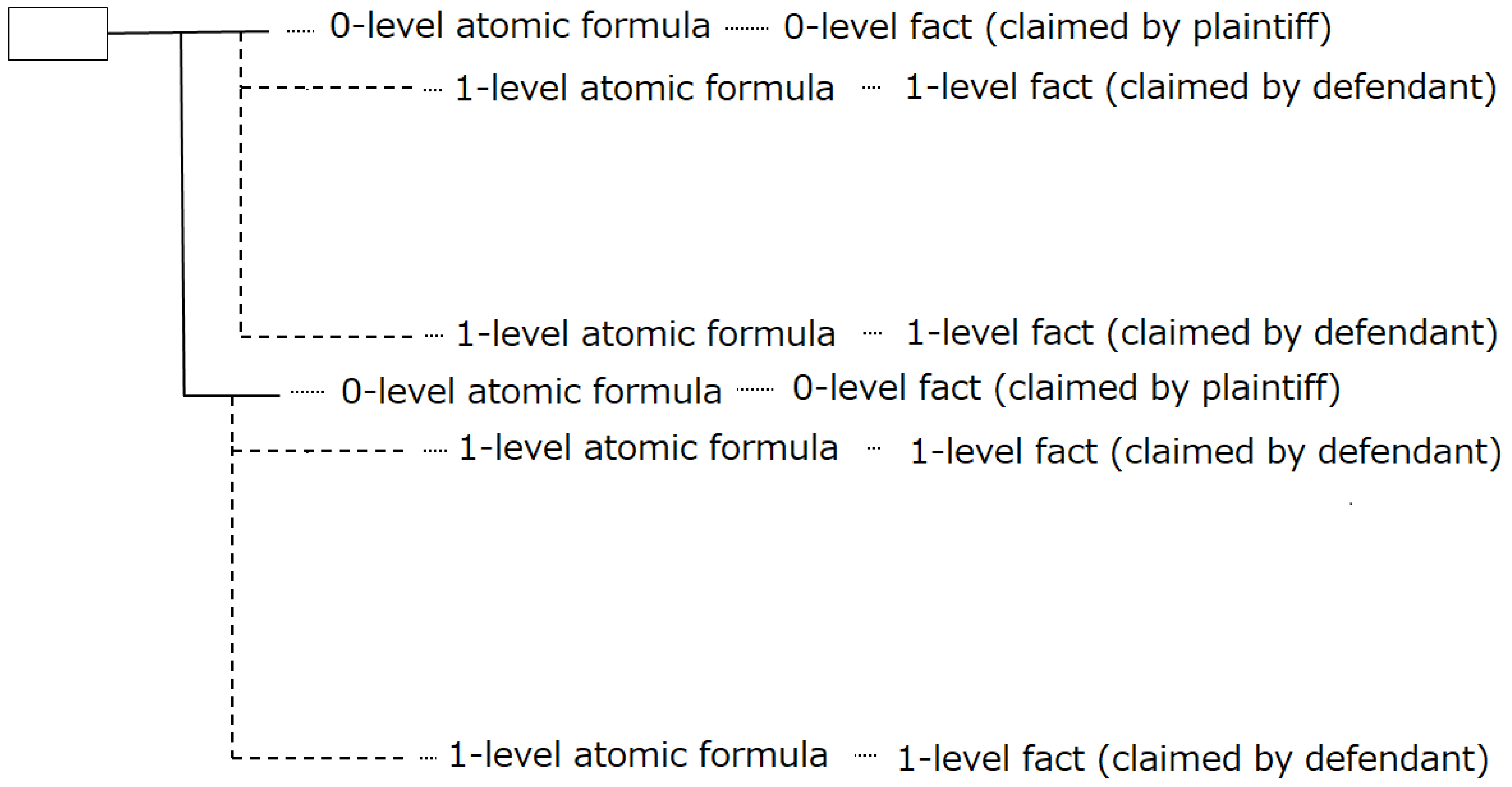
5. If all the concrete facts are entered, the modified proof tree is displayed.

6. Continue above until there are no argument raised from either side.

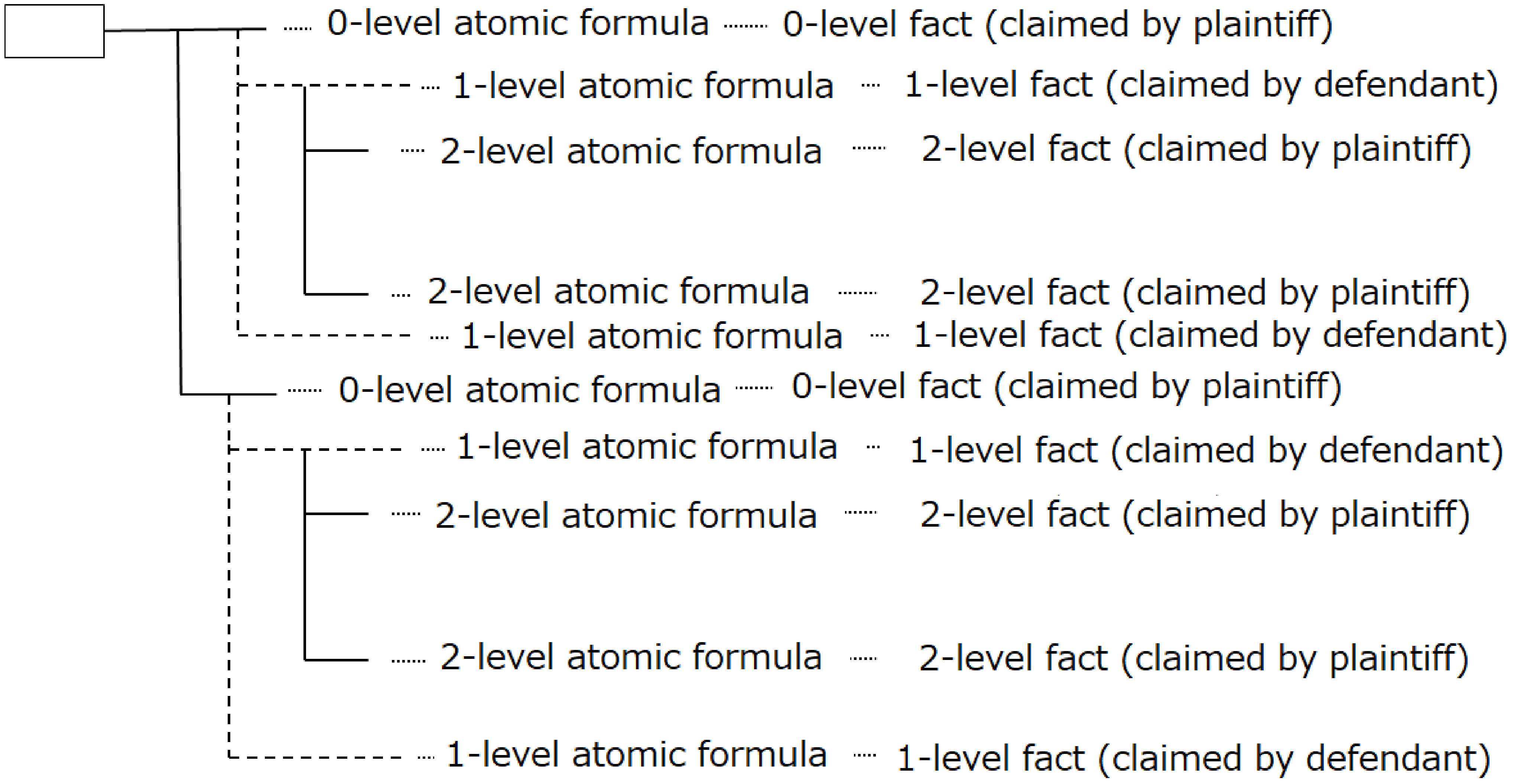
Block Diagram at Turn 0


0-level atomic formula
0-level fact (claimed by plaintiff)
0-level atomic formula
0-level fact (claimed by plaintiff)

Block Diagram at Turn 1


0-level atomic formula ······· 0-level fact (claimed by plaintiff)
1-level atomic formula ···· 1-level fact (claimed by defendant)
1-level atomic formula ···· 1-level fact (claimed by defendant)
0-level atomic formula ······· 0-level fact (claimed by plaintiff)
1-level atomic formula ···· 1-level fact (claimed by defendant)
1-level atomic formula ···· 1-level fact (claimed by defendant)

Block Diagram at Turn 2


0-level atomic formula
0-level fact (claimed by plaintiff)
1-level atomic formula
1-level fact (claimed by defendant)
2-level atomic formula
2-level fact (claimed by plaintiff)
2-level atomic formula
2-level fact (claimed by plaintiff)
1-level atomic formula
1-level fact (claimed by defendant)
0-level atomic formula
0-level fact (claimed by plaintiff)
1-level atomic formula
1-level fact (claimed by defendant)
2-level atomic formula
2-level fact (claimed by plaintiff)
2-level atomic formula
2-level fact (claimed by plaintiff)
1-level atomic formula
1-level fact (claimed by defendant)

Demonstration

You can see the demonstration at:

`http://research.nii.ac.jp/~ksatoh/PROLEGdemo/IssueArrangmentDemo.mp4`

Conclusion

- Extension of PROLEG to Arrange Issues.
  - Attorneys can prevent missing claims by using this system for a checking.
  - Lay person could make correct legal claims.
- Future Work
  - Evaluation of the System
  - Automatic Information Extraction from Claims written in a Natural Language