*ეძღვნება მამაჩემის, ერნაზი კურტანიძის ხსოვნას...*

# შინაარსი

# ანოტაცია

ავტომატური მსჯელობა ლოგიკისა და კომპიუტერულ მეცნიერებათა ერთ-ერთი ყველაზე მნიშვნელოვანი დარგია. ის ასევე განიხილება როგორც ხელოვნური ინტელექტის ქვედარგი. ეს მიმართულება სწავლობს მსჯელობის სხვადასხვა ასპექტებს. ავტომატური მსჯელობის ყველაზე მნიშვნელოვანი ატრიბუტებია კლასიკური პრედიკატთა აღრიცხვა და ლოგიკის სხვადასხვა კალკულუსები.

ჩვენ წარმოვადგინეთ მიზანზე-მიმართული მტკიცებათა ძებნის ალგორითმი სქემათა ფორმულებისთვის, რომელიც დაფუძნებულია სეკვენციათა კალკულუსზე. როგორც წესი სეკვენციათა კალკულუსის წესები შეიძლება თავისუფლად გამოყენებული იქნას დამტებითი ძებნის სივრცის შესაქმნელად. სტანდარტული მიდგომები ვრცელდება სქემათა ფორმულებისთვის, რათა თავიდან ავიცილოთ დამატებითი სივრცეები მტკიცებათა ძებნისას. სქემათა ფორმულები არის სასრული წარმოდგენა პირველი რიგის ფორმულების უსასრულო მიმდევრობის, ამგვარად ავტომატიზაციის პროცესის დასრულება არ არის მიზანშეწონილი, მიუხედავად ამისა, არსებობს ქვეკლასები (არა ტრივიალური), სადაც პროცესები არის სრულად ავტომატიზირებული.

კვლევის პროცესში ჩვენ შევისწავლეთ შესაბამისობა მტკიცებათა სქემებსა და ურანგო ლოგიკას შორის. მტკიცებათა სქემა არის ახალი ფორმალიზმი, ინდუქციური მსჯელობის ალტერნატივა, სადაც ქათ-ელიმინაციის თეორემა გამართულია. ურანგო ლოგიკა წარმოადგენს ძალიან მნიშვნელოვან ფორმალიზმს ცოდნის წარმოდგენისა და სემანტიკური ქსელისთვის. ჩვენ აღვწერეთ გარდაქმნა თუ როგორ შეიძლება ჩაიწეროს ურანო ლოგიკის წინადადება სქემათა ფორმულით.

ნაშრომში განხილულია ურანგო ტაბლო კალკულუსი, რომელიც ვრცელდება კლასიკური პირველი რიგის ტაბლო კალკულუსის ფორმულებისთვის ურანგო ტერმებზე. კალკულუსის სისწორისა

და სისრულის თეორიები დამტკიცებულია და წარმოდგენილია მისი შესაძლებლობები სემანტიკურ ქსელთან დაკავშირებულ პროგრამებში.

საბოლოოდ, ჩვენ შევისწავლეთ სკოლემიზაცია ურანგო ლოგიკაში კლასიკური პირველი-რიგის სემანტიკით. სკოლემიზაცია არის პირველი-რიგის ლოგიკის ფორმულის გარდაქმნა, რომელიც ფორმულიდან აგდებს ყველა ეგზისტენციალურ კვანტორს. ეს მეთოდი აუცილებელია მტკიცებათა თეორიაში და ავტომატურ მსჯელობებში, განსაკუთრებით უარყოფის პრინციპზე დაფუძნებულ კალკულუსში, როგორიცაა რეზოლუციური და ტაბლო კალკულუსები. ჩვენ განვავრცეთ სკოლემიზაციის პროცედურა ურანგო ფორმულებისთვის ასევე დავამტკიცეთ პროცედურის სისწორე და სისრულე.

# Abstract

Automated reasoning is one of the most important research area in logic and computer science. It is also considered as a sub-field of artificial intelligence. It studies different aspects of reasoning. The most important tools of automated reasoning are different calculi for classical logic.

we present a goal-directed proof-search algorithm for formula schemata, which is based on a sequent calculus. Usually, sequent calculus inference rules can be applied freely, producing a redundant search space. The standard approaches are extended to formula schemata to get rid of redundancy in a proof-search. A formula schema is a finite representation of an infinite sequence of first-order formulas, thus complete automation of the process is not feasible. Still, there are some (not so trivial) subclasses, where the process can be fully automated.

In the research process we study correspondence between proof schemata and unranked logics. Proof schemata is a new formalism, an alternative to inductive reasoning, where cut-elimination theorem holds. Unranked logics are very important formalisms used in knowledge representation and semantic web. We describe a transformation, how an unranked logic sentence can be encoded into a formula schema.

In this thesis unranked tableaux calculus is discussed, which extends the classical first order tableaux calculus for formulas over unranked terms. The correctness and completeness theorems of the calculus are proved and its expressive power in Web-related applications are illustrated.

Finally we study skolemization for unranked logics with classical first-order semantics. Skolemization is a transformation on first-order logic formulae, which removes all existential quantifiers from a formula. This technique is vital in proof theory and automated reasoning, especially for refutation based calculi, like res-

olution, tableaux, etc. Here we extend skolemization procedure to unranked formulae and prove that the procedure is sound and complete.

# მადლიერება

განსაკუთრებული მადლობა მინდა გადავუხადო პირველ რიგში ჩემს სამეცნიერო ხელმძღვანელს, ბატონ მიხეილ რუხაიას, რომ არა მისი ჩართულობა ამ პროცესში, ნამდვილად ვერ გავართმევდი თავს ასეთ მცირე დროში ამხელა სამუშაოს შესრულებას.

დიდი მადლობა მინდა ვუთხრა ჩემს სადოქტორო პროგრამის ხელმძღვანელს, ბატონ ჰამლეტ მელაძეს, ასევე ბატონ თემური კივილაძეს გაწეული მხარდაჭერისთვის, გვერდში დგომისთვის და დახმარებისთვის.

დიდი მადლობა მინდა გადავუხადო მთლიანად ჩვენი ფაკულტეტის ყველა პროფესორ-მასწავლებელს ჩემს სასწავლო პროცესში გაწეული შრომისთვის და მათ მიერ მოცემული ცოდნისათვის. მადლობა წმ.ანდრია პირველწოდებულის სახელობის საპატრიარქოს ქართულ უნივერსიტეტს, რომ შემიქმნა ყველანაირი პირობა სწავლისა და სამეცნიერო მუშაობისათვის.

საბოლოოდ მინდა მადლიერება გამოვხატო ჩემი ოჯახის, მეგობრების და განსაკუთრებით ჩემი მეუღლის მიმართ, რომელთა დახმარებით და გვერდში დგომით შევძელი მე ამ სამუშაოს შესრულება.

# შესავალი

ავტომატური მსჯელობა [RV01a, RV01b] ლოგიკისა და კომპიუტერულ მეცნიერებათა ერთ-ერთი ყველაზე მნიშვნელოვანი დარგია. ის ასევე განიხილება როგორც ხელოვნური ინტელექტის ქვედარგი. ეს მიმართულება სწავლობს მსჯელობის სხვადასხვა ასპექტებს. ავტომატური მსჯელობის ყველაზე მნიშვნელოვანი ატრიბუტებია კლასიკური პრედიკატთა აღრიცხვა და ლოგიკის სხვადასხვა კალკულუსები.

ეს დარგი სათავეს იღებს მეოცე საუკუნის დასაწყისიდან, რასელის სისტემით, რომელიც ეფუძნებოდა ტიპთა თეორიას [WR12]. მოგვიანებით, ალტერნატიული *პრედიკატთა აღრიცხვა* შემოღებულ იქნა ჰილბერტის მიერ [HA28] და ამ აღრიცხვის ფორმალიზაცია *ნატურალურ დედუქციასა* და *სეკვენციათა კალკულუსში* შემოთავაზებული იქნა გენცენის მიერ [Gen35a, Gen35b]. ამ სისტემებში ლოგიკური ოპერატორის მნიშვნელობა განმარტებულია გამოყვანის წესებით. გენცენის ნაშრომი ასევე მოიცავს სისრულისა და სისწორის თეორემებს ამ სისტემებისათვის და ასევე ძალიან ცნობილ ე.წ. *ქათ-ელიმინაციის* თეორემას, რომლიდანაც გამომდინარეობს, რომ ლოგიკურად შესრულებადი ფორმულებისათვის მტკიცებები შეიძლება მოიძებნოს მარტივი სტრატეგიით (ამ ფორმულის დაშლით).

თეორემათა ავტომატური მტკიცება [Pfe99] ავტომატური მსჯელობის ქვედარგია, რომელიც სათავეს იღებს რობინსონის რეზოლუციის პრინციპიდან [Rob65, Lei97]. ეს კალკულუსი ზუსტად შეესაბამება იმ მოთხოვნებს, რაც საჭიროა მათემატიკური თეორემების დამტკიცებისათვის კომპიუტერული პროგრამების საშუალებით. ამიტომაც, ძალიან ბევრი ავტომატური დამამტკიცებელი არსებობს, რომლებიც იყენებენ რეზოლუციის პრინციპს. ყველაზე მეტად სახელოვანი და წარმადული დამამტკიცებლები არიან: Prover9 [McC10], Vampire [RV02], E [Sch04b], SPASS [WDF+09], LEO-II [BTPF08], და ა.შ.

თეორემათა ავტომატური მტკიცება არ აღმოჩნდა საკმარისი საინტერესო

მათემატიკური დამტკიცებებისათვის, რომლებიც მაგალითად, შეიცავენ ინდუქციას. პრობლემა მდგომარეობს იმაში, რომ ინდუქციის წესი ვერ წარმოდგინდება პირველი რიგის ლოგიკაში. ის არის ე.წ. მეტა წესი, ან წესის სქემა, რომელიც უსასრულო მოდუს პონენსის წესს შეესაბამება. ამიტომ, პროგრამას დამტკიცების ძებნისას ინდუქციის წესის გამოსაყენებლად სჭირდება მომხმარებელთან ინტერაქცია. ამას მივყევართ ავტომატური მსჯელობის სხვა ქვედარგთან სახელწოდებით თეორემათა ინტერაქციული მტკიცება.

ინდუქციის აქსიომის ფორმალიზება ხდება მეორე და მაღალი რიგის ლოგიკებში, მაგრამ ამ ლოგიკებს აქვთ ისეთი დიდი გამომსახველობითი უნარი, რომ ისინი არასრულია. ამიტომაც მომხმარებლის ინტერაქცია აუცილებელია მტკიცებათა ძებნისას. მტკიცებათა ასისტენტები, როგორიცაა Isabelle [NPW02] და Coq [B+97] იძლევიან ძალიან კარგ საშუალებას ასეთი ინტერაქციისათვის.

უკანასკნელ წლებში ვითარდება ინდუქციური სისტემების ალტერნატივა, სახელწოდებით – მტკიცებათა სქემები, რომელიც გვერდს უვლის ინდუქციის აქსიომის გამოყენებას (იხილეთ [ACP09, ACP11, DLRW12, DLRW15, Ruk13]). არსებობს ტაბლო დამამტკიცებელის, სახელად RegSTAB, რეალიზაცია წინადადებათა სქემების შეზღუდული კლასისათვის, რომელსაც ქვია რეგულარული სქემები (იხილეთ [ACP10]). ამ კლასის ყოველი გაფართოება არის ამოუხსნადი და შესაბამისად, მტკიცებათა ძიება სრულად ავტომატიზებადი არ არის.

სხვა ძლიერი გამომსახველობის მქონე ფორმალიზმები არის ენები, რომლებიც დაფუძნებულია ურანგო ალფაბეტზე, ანუ ფუნქციონალურ ან/და პრედიკატულ სიმბოლოებს დაფიქსირებული ადგილიანობა არ აქვთ, ძირითადად მათი გამოყენების საკმაოდ ფართო სფეროს გამო. ასეთ ენების საშუალებით ბუნებრივად შეიძლება XML დოკუმენტების და მათზე ოპერაციების მოდელირება, კრიპტოგრაფიული პროტოკოლების და პარალელური პროცესების ანალიზი და სხვა. ასეთ ენებს ურანგო ენებს უწოდებენ.

ერთ-ერთი ყველაზე საინტერესო ფორმალიზმი, რაც ურანგო ალფაბეტს ეფუძნება, არის Common Logic [CLs07]. ეს არის ლოგიკური ენა, რომელიც განსხვავებულ სისტემებს და ქსელებს შორის ინფორმაციის გასაცვლელად შეიქმნა და რომელსაც 2007 წელს ISO/IEC-ის საერთაშორისო სტანდარტი მიენიჭა. Common Logic-ს, თუ შეიძლება ასე ითქვას, ენის მოქნილობა უკიდურესობამდე მიჰყავს: არაა ფორმალური განსხვავება პრედიკატულ და ფუნქციონალურ სიმბოლოებს შორის; სიმბოლოებს, რამდენიმე ლოგიკური სიმბოლოს გარდა, არ აქვთ ფიქსირებული ადგილიანობა; დაშვებულია ტერმების და მიმდევრობის ცვლადები; ტერმებმა შეიძლება ტერმებზე იმოქმედონ და ტერმები წარმოქმნან და ა.შ. ამ ყველაფრის მიუხედავად, რომელიც აღიქმება როგორც მაღალი რიგის ფუნქციონალობა, Common Logic-ს აქვს ზუსტად განსაზღვრული პირველი რიგის სემანტიკა, რაც სტანდარტშია აღწერილი.

Common Logic და მსგავსი ენები სულ უფრო და უფრო ხშირად გამოიყენება ცოდნის წარმოდგენისათვის. მათზე ონთოლოგიებიც იწერება, როგორიცაა, მაგალითად SUMO[1]. ყოველივე ეს ზრდის მოთხოვნას ისეთი გამოყვანის მეთოდების შექმნასა და გაუმჯობესებაზე, რამაც ხელი უნდა შეუწყოს ურანგო ენებში მსჯელობების ავტომატიზებას. თუმცა, ჩვენს ხელთ არსებული ინფორმაციით, გამოყვანის შესაბამისი სისტემები ჯერჯერობით ძალიან ცოტაა. ალბათ, [HV06] ერთადერთი ფართომასშტაბიანი მცდელობაა ურანგო ენებისთვის გამოყვანის სისტემის მისადაგებისა. უკანასკნელ წლებში, TPTP ბიბლიოთეკაში [Sut09] შესაბამისი პრობლემების დამატების შემდეგ, გააქტიურდა კვლევები, რაც ასეთი გამოყვანის სისტემების განვითარებას ისახავს მიზნად (მაგალითისათვის იხილეთ [BP10]).

წარმოდგენილი ნაშრომის მიზანია თეორემათა ავტომატური და ინტერაქციული მტკიცების შესწავლა სქემებისა და ურანგო ლოგიკისათვის. სქემები, რომლებსაც ჩვენ განვიხილავთ, პრიმიტიულ-რეკურსულ განსაზღვრებებზეა დაფუძნებული. ფორმულათა სინტაქსი გავრცობილია (სქემატური) ოპერატორებით $\bigwedge$ და $\bigvee$, რომლებიც მოქმედებენ $[a, b]$

[1] Suggested Upper Merged Ontology, `http://www.ontologyportal.org`

ინტერვალზე, სადაც $a, b$ უნდა შეიცავდეს არაუმეტეს ერთ პარამეტრს. სქემატური დამტკიცება არის ჩვეულებრივი სეკვენციათა კალკულუსის გამოყვანათა წყვილი, სადაც ერთი გამოყვანა მიესადაგება ინდუქციური განსაზღვრის საბაზისო ნაწილს, ხოლო მეორე რეკურსულ ნაწილს. საბაზისო ნაწილის ყველა აქსიომა უნდა იყოს ატომური და რეკურსულ ნაწილში, დამატებით, მტკიცებათა მიმმართველებია დაშვებული, რომლებიც ე.წ. ციკლურ მტკიცებებს ქმნიან. ასეთ მიმმართველებს *მტკიცებათა მაკავშირებელს* ვუწოდებთ, ხოლო თვითონ მტკიცებებს კი *მტკიცებათა სქემებს*.

ნაშრომის მიზნის მისაღწევად ჩვენ ავაგებთ მტკიცებათა სქემების ძებნის (ნახევრად-)ავტომატურ ალგორითმს. შემდეგ ჩვენ განვიხილავთ ურანგო ლოგიკასა და მის შესაბამისობას სქემებთან. რადგანაც კვლევები ორივე ტიპის ლოგიკაში ახალი მიმართულებებია, ჩვენს ხელთ არსებული ინფორმაციით, ჯერჯერობით არავის უფიქრია ამ ლოგიკებს შორის შესაბამისობის პოვნა. მაგრამ თუ უფრო ახლოს შევხედავთ სქემატურ ოპერატორებს, ისინი შეიძლება განხილულ იქნან როგორც ურანგო ოპერატორები. შესაბამისად, ჩვენ განვავრცობთ დამტკიცებათა ძებნის ალგორითმს ურანგო ლოგიკისათვის, აგრეთვე შემოვიღებთ ასეთი ლოგიკისათვის ტაბლო მეთოდს.

ნაშრომის დანარჩენი ნაწილი შემდეგნაირადაა ორგანიზებული: თავი 1-ში ჩვენ განვიხილავთ მტკიცებათა თეორიის საბაზისო ცნებებს, როგორიცაა სეკვენცია, სეკვენციათა კალკულუსი და სხვა.

თავი 2 შეეხება ფორმულათა სქემებს. ჩვენ განვიხილავთ სეკვენციათა კალკულუსს LKS და მასზე მტკიცებათა აგების ალგორითმს. მოვიყვანთ ამ ალგორითმის რეალიზაციის დეტალებს.

თავი 3 დაეთმობა ურანგო ლოგიკას. ჩვენ შემოვიღებთ სეკვენციათა კალკულუსს LKU და მასზე განვავრცობთ წინა თავში აღწერილ დამტკიცებათა ძებნის მეთოდს. აგრეთვე განხილული იქნება ტაბლო კალკულუსი ურანგო ლოგიკისათვის და მასთან დაკავშირებული ტექნიკები, როგორიცაა, მაგალითად, სკოლემიზაცია.

საბოლოოდ თავი 4 შეაჯამებს მიღებულ შედეგებს და ვისაუბრებთ სამომავლო გეგმებზე.

# თავი 1

# მტკიცებათა თეორიის ძირითადი ცნებები

მტკიცებათა თეორია სათავეს იღებს გენცენის შრომებიდან, როდესაც მან წარმოადგინა სეკვენციათა კალკულუსი, **L***ogische* **K***alkül*, პირველი რიგის ლოგიკისთვის [Tak87]. მას შემდეგ, ფორმალური დამტკიცებები ფართოდ გამოიყენება კომპიუტერულ მეცნიერებებში, კერძოდ, პროგრამებისა და აპარატურის შემოწმებისათვის. ყოველივე ამან დასაბამი მისცა თეორემათა ავტომატური მტკიცების დარგს, რაც არის მათემატიკური ლოგიკის განშტოება. არსებობს სხვადასვა ტექნოლოგიები თეორემათა დამტკიცებისათვის, როგორიცაა რეზოლუციის მეთოდი, ტაბლო მეთოდი და სხვა. ქვემოთ ჩვენ განვიხილავთ მხოლოდ სეკვენციათა კალკულუსს. ტაბლო მეთოდი ურანგო ლოგიკისათვის განხილული იქნება თავი 3-ში.

## 1.1 სეკვენციათა კალკულუსი LK

ჩვენ განვიხილავთ პირველი რიგის ენას [Tak87]-ის მიხედვით, რომელიც შედგება ცვლადთა სიმრავლისგან, $n$-ური ფუნქციუნალური და პრედიკატული სიმბოლოებისაგან. *ტერმები* აგებულია სტანდარტული ინდუქციური გზით ცვლადებისა და ფუნქციონალური სიმბოლოების გამოყენებით. *ფორმულები* აგებულია ინდუქციური გზით ატომების და ლოგიკური ოპერატორიების $\neg, \wedge, \vee, \Rightarrow, \forall$ და $\exists$ გამოყენებით. ფორმულაში ცვლადთა შემოსვლას ეწოდება *დაბმული* თუ ისინი არიან $\forall$ ან $\exists$ კვანტორების საზღვრებში, სხვა შემთხვევაში ეწოდებათ *თავისუფალი*. ფორმულის ინტერპრეტაციის, შესრულებადობის და ზოგადმართებულობის ცნებები განსაზღვრულია

სტანდარტული კლასიკური აზრით.

ატომები წარმოადგენს საკუთარი თავის *ქვეფორმულებს*, ხოლო ფორმულების $\neg A, A \bullet B$ და $(Qx)A(x)$ ქვეფორმულები, სადაც $\bullet \in \{\wedge, \vee, \Rightarrow\}$ და $Q \in \{\forall, \exists\}$, წარმოადგენენ $A$-ს, $A$-სა და $B$-ს და $A(t)$-ს ქვეფორმულებს, $t$ ნებისმიერი ტერმისათვის, შესაბამისად, და თითონ ამ ფორმულებს.

**განსაზღვრება 1.1.1** (სეკვენცია)**.** გამოსახულებას $\Gamma \vdash \Delta$, სადაც $\Gamma$ და $\Delta$ არიან სასრული (შესაძლებელია ცარიელი) ფორმულათა სიმრავლეები, ეწოდება *სეკვენცია*. $\Gamma$ არის სეკვენციის *წინაპირობა* და $\Delta$ არის *შედეგი*. ორი სეკვენცია არის *ექვივალენტური*, თუ სეკვენციების წინაპირობებისგან და შედეგებისგან მიღებული სიმრავლეები შესაბამისად ტოლია.

სეკცენციაში შემავალი ფორმულების ქვეფორმულებს ეწოდებათ სეკვენციათა ქვეფორმულები. ჩვენ ვიყენებთ ბერძნული ანბანის ასოებს $\Gamma, \Delta, \Pi, \Lambda$ რომ აღვნიშნოთ სასრული (შეიძლება ცარიელი) ფორმულათა სიმრავლეები.

სემანტიკურად, სეკვენცია $S : A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ ექვივალენტურია ფორმულის $F : (A_1 \wedge \cdots \wedge A_n) \Rightarrow (B_1 \vee \cdots \vee B_m)$. ვამბობთ, რომ $I$ არის $S$-ის ინტერპრეტაცია, თუ ის არის $F$-ის ინტერპრეტაცია. $S$ ზოგადმართებულია მაშინ და მხოლოდ მაშინ, როდესაც $F$ ზოგადმართებულია. ცარიელი სეკვენცია $\vdash$ წარმოადგენს ფორმულას $\top \Rightarrow \bot$, რაც არის $\bot$ (მცდარი).

**განსაზღვრება 1.1.2** (გამოყვანის წესები)**.** გამოყვანის წესი წარმოადგენს გამოსახულებას

$$\frac{S_1}{S} \quad \text{ან} \quad \frac{S_1 \quad S_2}{S}$$

სადაც $S_1$ და $S_2$ ეწოდებათ ზედა სეკვენცია, ხოლო $S$ ეწოდება ქვედა სეკვენცია. ფორმულებს, რომლებზეც წესები მოქმედებს, ეწოდება *დამხმარე ფორმულები* და ფორმულას რომელიც გამოჰყავს წესს ეწოდება *მთავარი ფორმულა*. დამხმარე ფორმულები წარმოადგენს მთავარი ფორმულის *წინარე* ფორმულას და ეს მიმართება არის ტრანზიტიული.

**განსაზღვრება 1.1.3** (საწყისი სეკვენცია)**.** $A \vdash A$ ფორმის სეკვენციას, სადაც $A$ არის ატომალური ფორმულა, ეწოდება საწყისი სეკვენცია.

**განსაზღვრება 1.1.4** (LK კლალკულუსი)**.** სეკვენციათა კალკულუსი LK შეიცავს საწყის სეკვენციებს აქსიომების სახით, და შედგება შემდეგი წესებისაგან:

1. ლოგიკური წესები:

   - $\neg$ შემოტანა

     $$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}\neg\colon l \quad \text{და} \quad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\neg\colon r$$

   - $\wedge$ შემოტანა

     $$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\wedge\colon l \quad \text{და} \quad \frac{\Gamma \vdash \Delta, A \quad \Pi \vdash \Lambda, B}{\Gamma, \Pi \vdash \Delta, \Lambda, A \wedge B}\wedge\colon r$$

   - $\vee$ შემოტანა

     $$\frac{A, \Gamma \vdash \Delta \quad B, \Pi \vdash \Lambda}{A \vee B, \Gamma, \Pi \vdash \Delta, \Lambda}\vee\colon l \quad \text{და} \quad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}\vee\colon r$$

   - $\Rightarrow$ შემოტანა

     $$\frac{\Gamma \vdash \Delta, A \quad B, \Pi \vdash \Lambda}{A \Rightarrow B, \Gamma, \Pi \vdash \Delta, \Lambda}\Rightarrow\colon l \quad \text{და} \quad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B}\Rightarrow\colon r$$

2. კვანტორული წესები:

   - $\forall$ შემოტანა

     $$\frac{A(t), \Gamma \vdash \Delta}{(\forall x)A(x), \Gamma \vdash \Delta}\forall\colon l \quad \text{და} \quad \frac{\Gamma \vdash \Delta, A(u)}{\Gamma \vdash \Delta, (\forall x)A(x)}\forall\colon r$$

     სადაც $t$ არის ნებისმიერი ტერმი და $u$ არის თავისუფალი ცვლადი რომელიც არ გვხვდება ქვედა სეკვენციაში; $u$-ს ეწოდება *უნიკალური ცვლადი* და პირობას, რომ ეს ცვლადი არ უნდა იყოს ქვედა სეკვენციაში, ეწოდება *უნიკალური ცვლადის პირობა*. $\forall\colon l$ წარმოადგენს სუსტ კვანტორულ წესს, ხოლო $\forall\colon r$ - ძლიერ კვანტორულ წესს.

   - $\exists$ შემოტანა

     $$\frac{A(u), \Gamma \vdash \Delta}{(\exists x)A(x), \Gamma \vdash \Delta}\exists\colon l \quad \text{და} \quad \frac{\Gamma \vdash \Delta, A(t)}{\Gamma \vdash \Delta, (\exists x)A(x)}\exists\colon r$$

სადაც $t$ არის ნებისმიერი ტერმი და $u$ არის უნიკალური ცვლადი, რომელიც აკმაყოფილებს უნიკალური ცვლადის პირობას. $\exists{:}\, l$ ეწოდება ძლიერი კვანტორული წესი, ხოლო $\exists{:}\, r$ - სუსტი კვანტორული წესი.

3. სტრუქტურული წესები:

   - შესუსტების წესები:

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} w{:}\, l \quad \text{და} \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A} w{:}\, r$$

   - შეკვეცის წესი:

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} c{:}\, l \quad \text{და} \quad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A} c{:}\, r$$

   - განკვეთის წესი:

$$\frac{\Gamma \vdash \Delta, A \quad A, \Pi \vdash \Lambda}{\Gamma, \Pi \vdash \Delta, \Lambda} cut$$

**შენიშვნა 1.1.5.** მას შემდეგ, რაც ჩვენ განვსაზღვრეთ სეკვენცია სიმრავლეების საშუალებით, ჩვენ აღარ გვჭირდება გადანაცვლების წესები.

**განსაზღვრება 1.1.6** (LK-გამოყვანა)**.** LK-გამოყვანა $\phi$ ეს არის ხე, რომლის კვანძებზეც არის სეკვენცია და ტოტები არის გამოყვანის წესები. ხის ფუძეში არსებულ სეკვენცია $S$-ს ეწოდება საბოლოო სეკვენცია და იწერება ბოლოში. ვთქვათ, $\mathcal{A}$ არის სეკვენციათა სიმრავლე $\phi$-ის ტოტების წვეროებზე, მაშინ შეგვიძლია ვთქვათ, რომ $\phi$ არის $S$-ის გამოყვანა $\mathcal{A}$-დან (აღინიშნება $\mathcal{A} \vdash_{\mathsf{LK}} S$).

**განსაზღვრება 1.1.7** (LK-დამტკიცება და LK-ქვემტკიცება)**.** LK-დამტკიცება არის LK-გამოყვანა, რომლის წვეროებზე გვაქვს საწყისი სეკვენციები. თუ $\phi$ არის $\Gamma \vdash \Delta$ საბოლოო სეკვენციის LK-დამტკიცება, მაშინ შეგვიძლია დავწეროთ:

$$\begin{array}{c} \phi \\ \Gamma \vdash \Delta \end{array}$$

და ვამბობთ, რომ $\Gamma \vdash \Delta$ არის დამტკიცებადი LK-ში. LK-დამტკიცებას განკვეთის წესის გარეშე ეწოდება *განკვეთის წესისაგან თავისუფალი*. $\phi$-ს LK-ქვემტკიცება არის $\phi$-ს ნებისმიერი ქვე-ხე, რომელიც აგრეთვე არის LK-დამტკიცება.

**წინადადება 1.1.1.** LK *კალკულუსი არის სწორი და სრული, ანუ სეკვენცია* $S$ *გამოყვანადია* LK*-ში მაშინ და მხოლოდ მაშინ, როდესაც* $S$ *ზოგადმართებულია.*

*დამტკიცება.* იხილეთ [Tak87]-ში. □

# თავი 2

# მტკიცებათა სქემები და მათი ავტომატური აგება

როგორც წინა თავში ვიხილეთ, პირველი რიგის ლოგიკა არ არის გადაწყვეტადი, შესაბამისად, ნებისმიერი სრული დამტკიცებათა ძებნის პროცედურა არის არადასრულებადი.

თერმათა სქემატიზაციის კონცეფცია შემოღებულ იქნა [CHK91]-ში რათა თავიდან აეცილებინათ არადასრულებადობა სიმბოლური გამოთვლებისას და წარმოედგინათ არასასრული გამოყვანები სასრული ჩანაწერით. მოგვიანებით, [ACP09]-ში, ფორმულათა სქემები შეიქმნა წინადადებათა ლოგიკისათვის ისეთ სქემატურ პრობლემებთან უკეთ მუშაობისათვის, როგორიცაა, მაგალითად, დიაგრამების გაფერადება, ციფრული წრედები და სხვა. ეს ენა იმდენად მძლავრია, რომ შესრულებადობის პრობლემა გადაწყვეტადი არ არის [ACP11]. მიუხედავად ამისა, არსებობს გადაწყვეტადი ფრაგმენტები და ამ ფრაგმენტებისათვის ტაბლო დამამტკიცებელი [ACP10].

[DLRW15, Ruk13]-ში ფორმულათა სქემების ენა განვრცობილია პირველი რიგისათვის, რომელიც საშალებას გვაძლევს პირველი რიგის ფორმულების უსასრულო სიმრავლე წარმოვადგინოთ სასრული სახით. სეკვენციათა კალკულუსი, სახელწოდებით LKS, იქნა განსაზღვრული, რომელიც შეიძლება ჩაითვალოს, როგორც ალტერნატივა სტანდარტული სეკვენციათა კალკულუსისა ინდუქციურ წესთან ერთად.

ჩვენი მიზანია, რომ განვსაზღვროთ ეფექტური მტკიცებათა ძებნის პროცედურა, რომელიც მოცემული პირველი რიგის ფორმულათა სქემისათვის ააგებს მის დამტკიცებათა სქემას. ცნობილია, რომ დამტკიცებათა ძებნის

უბრალო პროცედურები სეკვენციათა კალკულუსში არაეფექტურია, რაც გამოიხატება იმაში, რომ მივყავართ ჭარბ საძიებო სივრცესთან. მიზეზი არის ის, რომ წესები შეიძლება იყოს გამოყენებული არადეტერმინისტული გზით. მსგავსი შემთხვევების თავიდან ასაცილებლად ჟირარდმა წარმოადგინა პოლარობის კონცეფცია და მასზე დაფუძნებული ფოკუსირებული სეკვენციათა კალკულუსი LC [Gir91]. მოგვიანებით, [LM09]-ში LC ადაპტირებული იქნა დამტკიცებათა ძებნისათვის და მიღებულ იქნა სეკვენციათა კალკულუსი LKF. [BMS10]-ში კი ფოკუსირების ტექნიკა გამოყენებულ იქნა ინდუქციური თეორემების დასამტკიცებლად.

ფოკუსირების იდეა მდგომარეობს ლოგიკური კავშირების კლასიფიკაციაში სინქრონულ და ასინქრონულ პოლუსებად. ეს პოლუსები არ მოქმედებს ფორმულების დამტკიცებადობაზე, მაგრამ გვაჩვენებს მტკიცებათა აგების ნაბიჯებს. ჩვეულებრივ, ასინქრონული წესები ინვერსულია და შეიძლება იქნეს გამოყენებული ნებისმიერი თანმიმდევრობით. ამის საპირისპიროდ, თუ სინქრონული ფორმულა არჩეულია დასაშლელად, იწყება სინქრონული ფაზა და ყველა სინქრონული ქვეფორმულა უნდა დაიშალოს, სანამ აქსიმები ან მხოლოდ ასინქრონული ქვეფორმულები არ დაგვრჩება. გენცენის სეკვენციათა კალკულუსისაგან განსახვავებით, ამ ტექნიკის გამოყენებით მტკიცებათა აგების ნაბიჯები ხდება დეტერმინირებადი.

ჩვენი მიზანი არ არის ფოკუსირებული სეკვენციათა კალკულუსის აგება ფორმულათა სქემებისათვის, მაგრამ განვიხილავთ LKS კალკულუსის ინვერსულ ვერსიას, სადაც გამოვიყენებთ ფოკუსირების ტექნიკის პრინციპებს, რომ შევამციროთ დამტკიცებათა ძების არე და უფრო ეფექტური გავხადოთ პროცედურა. ასევე, დამტკიცებათა ძებნისას წინა მდგომარეობებში დაბრუნების თავიდან ასაცილებლად გამოვიყენებთ [AMS98]-ში მოცემულ მიდგომას.

## 2.1 პროპოზიციული ფორმულათა სქემები

პროპოზიციული ფორმულათა სქემები პირველად წარმოდგენილი იყო [ACP09]-ში; მისი ქვეკლასი, ე.წ *რეგულარული სქემები* გამოკვლეული იქნა და დამტკიცდა, რომ ფორმულათა სქემების ეს კლასი გადაწყვეტადია. მოგვიანებით [ACP11]-ში განსაზღვრული იქნა ახალი ქვეკლასი სახელად *წრფივად შემოსაზღვრული სქემები* და მოცემული იქნა რეგულარული სქემებზე დაყვანის ალგორითმი. აქედან გამომდინარე, დამტკიცებული იქნა, რომ წრფივად შემოსაზღვრულ სქემათა კლასიც გადაწყვეტადია. აქ განსაზღვრულია წრფივად შემოსაზღვრული სქემის მცირედით მოდიფიცირებული ვერსია, მაგრამ მისი გამომსახველობითი უნარი (ანუ გადაწყვეტადობა) შენარჩუნებულია.

### 2.1.1 სინტაქსი

ჩვენ განვიხილავთ ინდექსის ცვლადების სასრულ სიმრავლეს (რომლებიც წარმოდგენილი იქნება $\mathbb{N}$ სიმრავლეზე), და პროპოზიციული სიმბოლოების საასრულ სიმრავლეს. შემდგომში, ამ სქციაში ჩვენ ვნახავთ, რომ ინდექსის ცვლადები შესაძლებელია იყოს როგორც თავისუფალი ასევე დაბმული. თავისუფალ ინდექსის ცვლადებს ეწოდება *პარამეტრი*. *წრფივი არითმეტიკული გამოსახულების* სიმრავლე ან მარტივად *არითმეტიკული გამოსახულება* სტანდარტულად განსაზღვრულია მუდმივა 0-ით, ინდექსის ცვლადებითა და $s, +$ ფუნქციონალური სიმბოლოებით (ნაგულისხმევია $+$-ის სტანდარტული თვისებები, მაგ.: $s(0) + s(0)$ ექვივალენტურია $s(s(0))$). ხშირად ნატურალურ რიცხვებს $\alpha \in \mathbb{N}$ გამოვსახავთ $s^{\alpha}(0)$ გამოსახულებით და პირიქით. თუ $n$ არის ინდექსის ცვლადი, მაშინ $n + \cdots + n$ ($\alpha$ -ჯერ) ჩაიწერება $\alpha.n$. ასევე ჩვენ აღვნიშნავთ:

- $\mathbb{N}$-ის ელემენტებს: $\alpha, \beta, ...$,
- დაბმულ ინდექსის ცვლადებს : $i, j, l, ...$,
- პარამეტრებს: $k, m, n, ...$,

- წრფივ არითმეტიკულ გამოსახულებებს: $a, b, ...,$
- პროპოზიციულ სიმბოლოებს: $p, q, ....$

არითმეტიკული გამოსახულება $a$ არის *შემოსაზღვრული* თუ $a$ არ შეიცავს ინდექსის ცვლადებს. ცხადია, შემოსაზღვრული არითმეტიკული გამოსახულება შეიძლება გადაიწეროს რიცხვების სახით, ამიტომ ჩვენ შეგვიძლია ისინი განვიხილოთ მათი ნორმალური ფორმებით, რაც ემთხვევა ნატურალურ რიცხვებს.

*ჩასმაა* არის ფუნქცია, რომელიც ინდექსის ცვლადებს ასახავს არითმეტიკულ გამოსახულებაში. $k$ ინდექსის ცვლადის ჩანაცვლება $a$ არითმეტიკული გამოსახულებიით აღინიშნება $[k/a]$.

**განსაზღვრება 2.1.1** (ინდექსირებული წინადადება)**.** გამოსახულებას $p_a$, სადაც $a$ არის არითმეტიკული გამოსახულება, ხოლო $p$ კი პროპოზიციული სიმბოლო, ეწოდება *ინდექსირებული წინადადება*. თუ $a$ არის შემოსაზღვრული, მაშინ ჩვენ შეგვიძლია განვიხილოთ შემოსაზღვრული ინდექსირებული წინადადებები, რომლებსაც ვუწოდებთ *პროპოზიციულ ცვლადებს*.

**განსაზღვრება 2.1.2** (ფორმულათა სქემა)**.** ფორმულათა სქემა განისაზღვრება ინდუქციური გზით შემდეგნაირად:

- ინდექსირებული წინადადება არის ფორმულათა სქემა (ატომი).
- თუ $A$ და $B$ არიან ფორმულათა სქემები, მაშინ $\neg A$, $A \vee B$, $A \wedge B$ და $A \Rightarrow B$ აგრეთვე არიან ფორმულათა სქემები.
- თუ $A$ არის ფორმულათა სქემა, ხოლო $a, b$ კი არითმეტიკული გამოსახულებები და $i$ - ინდექსის ცვლადი, რომელიც არ არის დაბმული $A$ ფორმულაში, მაშინ $\bigwedge_{i=a}^{b} A$ და $\bigvee_{i=a}^{b} A$ არიან ფორმულათა სქემები, სადაც $i$ ცვლადი დაბმულია ორივე ფორმულათა სქემაში.

ჩვენ აღვნიშნავთ ფორმულათა სქემებს $A, B, ....$ $\sigma$ ჩანაცვლების $A$ ფორმულათა სქემაზე მისადაგების ცნება განსაზღვრულია სტანდარტულად

და აღინიშნება $A\sigma$-თი. ჩანაწერი $A(k)$ გამოიყენება, $A$ ფორმულათა სქემაში $k$ პარამეტრის იდენტიფიცირებისათვის, ხოლო ჩანაწერი $A(a)$ აღნიშნავს ფორმულას $A[k/a]$.

**შენიშვნა 2.1.3.** ჩვენ შემოვიღეთ $\Rightarrow$ მეტი მოსახერხებლობისათვის და დავუშვებთ, რომ $\Rightarrow$ არის მარჯვნივ ასოციაციური; მაშინ, იმლიკაციათა ჯაჭვი $A(0) \Rightarrow A(1) \Rightarrow \cdots \Rightarrow A(n+1)$ შეიძლება წარმოვადგინოთ, როგორც $(\bigvee_{i=0}^{n} \neg A(i)) \vee A(n+1)$ ან $(\neg \bigwedge_{i=0}^{n} A(i)) \vee A(n+1)$. შევნიშნოთ, რომ ფორმულის $(((A(0) \Rightarrow A(1)) \Rightarrow A(2)) \cdots) \Rightarrow A(n+1)$ პირდაპირ ჩაწერა ჩვენს ფორმალიზმში შეუძლებელია, მაგრამ ამ ფორმულის შესრულებადად-ექვივალენტური ფორმულის კოდირება შესაძლებელია.

**განსაზღვრება 2.1.4** (ქვეფორმულები)**.** ქვეფორმულები განსაზღვრულია შემდგენაირად:

- ატომალური ფორმულა არის თავისი თავის ქვეფორმულა.
- ფორმულა $\neg A$-ს ქვეფორმულებია $A$-ს ქვეფორმულები და თვითონ $\neg A$.
- ფორმულების $A \vee B$, $A \wedge B$ და $A \Rightarrow B$ ქვეფორმულებია: $A$ და $B$-ს ქვეფორმულები და თავიანთი თავი.
- ფორმულის $\bigwedge_{i=a}^{b+1} A(i)$ ქვეფორმულებია: $A(b+1)$ და $\bigwedge_{i=a}^{b} A(i)$-ს ქვეფორმულები და თავისი თავი.
- ფორმულის $\bigvee_{i=a}^{b+1} A(i)$ ქვეფორმულებია: $A(b+1)$ და $\bigvee_{i=a}^{b} A(i)$-ს ქვეფორმულები და თავისი თავი.

ფორმულათა სქემა $A$ არის *წრფივად შემოსაზღვრული* [ACP11] მაშინ და მხოლოდ მაშინ, როდესაც სრულდება შემდეგი პირობები:

1. $A$ შეიცავს არაუმეტეს ერთ $n$ პარამეტრს.
2. ყოველი ინდექსირებული წინადადება $A$ არის შემდეგი ფორმის: $p_{\alpha.n+\beta.i+\gamma}$, სადაც $n$ არის პარამეტრი, $i$ დაბმული ცვლადი, $\alpha, \gamma$ კი ნებისმიერი ნატურალური რიცხვები და $\beta \in \{0, 1\}$.

3. თუ $A$ შეიცავს იტერაციას $\bigwedge_{i=a}^{b} B$ (ან $\bigvee_{i=a}^{b} B$) მაშინ, $a, b$ არიან შესაბამისად შემდეგი ფორმის: $\alpha.n + \beta$ და $\gamma.n + \kappa.j + \iota$, სადაც $\alpha, \beta, \gamma, \iota$ არის ნებისმიერი ნატურალური რიცხვი, $\kappa \in \{0, 1\}$ და $j$ არის დაბმული ცვლადი.

არაფორმალურად რომ განვმარტოთ, ინდექსები და იტერაციის საზღვრები უნდა შეიცავდნენ არაუმეტეს ერთ პარამეტრსა და არაუმეტეს ერთ დაბმულ ცვლადს კოეფიციენტით 1.

### 2.1.2 სემანტიკა

ინტერპრეტაცია არის ფუნქციათა წყვილი $I = (\mathcal{I}, \mathcal{I}_p)$, ისეთი, რომ $\mathcal{I}$ არის ასახვა ინდექსის ცვლადებისა ნატურალურ რიცხვებში და $\mathcal{I}_p$ არის ასახვა პროპოზოციული ცვლადებისა ჭეშმარიტობის მნიშვნელობაში. ვთქვათ, $\mathcal{I}(a)$ არის $\mathcal{I}$-ს ჰომომორფული გაფართოება, არითმეტიკული $a$ გამოსახულებისთვის. ვთქვათ $\sigma$ არის ჩასმა, მაშინ $I\sigma$-თი ჩვენ აღვნიშნავთ ინტერპრეტაციას და ჩავწერთ შემდეგნაირად: $\mathcal{I}_p\sigma = \mathcal{I}_p$ და $\mathcal{I}\sigma(n) = \mathcal{I}(n\sigma)$ ყოველი ინდექსიანი $n$ ცვლადისთვის.

ჭეშმარიტობის მნიშვნელობა $[\![A]\!]_I$ ფორმულათა სქემისათვის $A$ ინტერპრეტაციაში $I$ განსაზღვრულია ინდუქციურად:

- $[\![p_a]\!]_I = \mathcal{I}_p(p_{\mathcal{I}(a)})$.
- $[\![\neg A]\!]_I = \mathbf{T}$ მაშინ და მხოლოდ მაშინ, როდესაც $[\![A]\!]_I = \mathbf{F}$.
- $[\![A \vee B]\!]_I = \mathbf{T}$ მაშინ და მხოლოდ მაშინ, როდესაც $[\![A]\!]_I = \mathbf{T}$ ან $[\![B]\!]_I = \mathbf{T}$.
- $[\![A \wedge B]\!]_I = \mathbf{T}$ მაშინ და მხოლოდ მაშინ, როდესაც $[\![A]\!]_I = \mathbf{T}$ და $[\![B]\!]_I = \mathbf{T}$.
- $[\![A \Rightarrow B]\!]_I = \mathbf{T}$ მაშინ და მხოლოდ მაშინ, როდესაც $[\![A]\!]_I = \mathbf{F}$ ან $[\![B]\!]_I = \mathbf{T}$.
- $\left[\!\left[\bigwedge_{i=a}^{b} A\right]\!\right]_I = \mathbf{T}$ მაშინ და მხოლოდ მაშინ, როდესაც ყოველი ნატურალური რიცხვისთვის $\alpha$, სადაც $\mathcal{I}(a) \leq \alpha \leq \mathcal{I}(b)$, $[\![A]\!]_{I[i/\bar{\alpha}]} = \mathbf{T}$, სადაც $\bar{\alpha}$ რიცხობრივად შეესაბამება $\alpha$-ს.

- $\left[\!\left[\bigvee_{i=a}^{b} A\right]\!\right]_I = \mathbf{T}$ მაშინ და მხოლოდ მაშინ, როდესაც არსებობს ნატურალური რიცხვი $\alpha$, ისეთი, რომ $\mathcal{I}(a) \leq \alpha \leq \mathcal{I}(b)$, $[\![A]\!]_{I[i/\bar{\alpha}]} = \mathbf{T}$, სადაც $\bar{\alpha}$ რიცხობრივად შეესაბამება $\alpha$-ს.

**შენიშვნა 2.1.5.** თუ $\mathcal{I}(a) > \mathcal{I}(b)$, მაშინ $\bigwedge_{i=a}^{b} A$ ყოველთვის არის ჭეშმარიტი და $\bigvee_{i=a}^{b} A$ ყოველთვის მცდარია ნებისმიერი ფორმულათა სქემა $A$-სთვის.

ფორმულათა სქემა $A$ შესრულებადია მაშინ და მხოლოდ მაშინ, თუ გვაქვს ინტერპრეტაცია $I$, ისეთი, რომ $[\![A]\!]_I = \mathbf{T}$, სხვა შემთხვევაში $A$ არ არის შესრულებადი. თუ ინტერპრეტაცია $I$ ასრულებს $A$, მაშინ ჩვენ ვწერთ : $I \models A$.

თუ ფორმულა არ შეიცავს თავისუფალ ინდექსის ცვლადებს ანუ პარამეტრებს, მაშინ ვიტყვით რომ ფორმულათა სქემა არის შემოსაზღვრული. მაშინ ცხადია, რომ შემოსაზღვრული ფორმულათა სქმეა არის ჩვეულებრივ პროპოზიციული ფორმულა (შემოსაზღვრული იტერაცია ექვივალენტურია პროპოზიციული ცვლადების სასრული გაერთიანება/თანაკვეთის).

**წინადადება 2.1.1.** *შესრულებადობის პრობლემა ნახევარად გადაწყვეტადია ფორმულათა სქემებში.*

*დამტკიცება.* ვთქვათ $A$ არის ფორმულათა სქემა, მაშინ ყოველი ინტერპრეტაციისათვის $I$, $I \models A$ მაშინ და მხოლოდ მაშინ, როდესაც არსებობს შემოსაზღვრული ჩასმა $\sigma$, ისეთი, რომ $I \models A\sigma$. რადგან $A\sigma$ არის შემოსაზღვრული ფორმულათა სქემა, ანუ მხოლოდ პროპოზიციული ფორმულა, $A\sigma$-ს შესრულებადობა არის გადაწყვეტადი. აგრეთვე, შემოსაზღვრული ჩასმების სიმრავლე არის რეკურსულად გადათვლადი. შესაბამისად, $A$-ს შესრულებადობის შემოწმების პროცედურა ხდება ნახევრად გადაწყვეტადი. □

ქვემოთ ჩვენ განვსაზღვრავთ წრფივად-შემოსაზღვრული სქემების ქვეკლასს, რომელსაც ვუწოდებთ *რეგულარულს* [ACP11]. ფორმულათა სქემა $A$ რეგულარულია თუ სრულდება შემდეგი პირობები:

1. $A$ შეიცავს არაუმეტეს ერთ პარამეტრ $n$-ს.

2. თუ $A$ შეიცავს იტერაციას $\bigwedge_{i=a}^{b} B$ (ან $\bigvee_{i=a}^{b} B$), მაშინ $a, b$ არიან შესაბამისად $\alpha$ და $n + \beta$ ფორმის, სადაც $\alpha, \beta \in \mathbb{N}$, $B$ არ შეიცავს რაიმე იტერაციას და ყოველ ინდექსირებულ წინადადებას $B$-ში აქვს ფორმა $p_{i+\gamma}$, სადაც $\gamma \in \mathbb{N}$.

3. ყველა იტერაციას $A$-ში აქვს ერთნაირი საზღვრები.

**წინადადება 2.1.2.** *შესრულებადობის პრობლემა გადაწყვეტადია წრფივრად-შემოსაზღვრული სქემებისთვის.*

*დამტკიცება (მონახაზი).* დამტკიცების იდეა არის წრფივად-შემოსაზღვრული ფორმულათა სქემის დაყვანა რეგულარულ სქმებზე. ეს რომ გავაკეთოთ საჭიროა მივყვეთ შემდეგ ნაბიჯებს:

1. მოვაშოროთ ჩადგმული იტერაციები.

2. ყველა იტერაცია უნდა გარდავქმნათ ისეთ იტერაციად, რომელიც იქნება მოცემული ინტერვალში $[\alpha, n + \beta],\ \alpha, \beta \in \mathbb{N}$.

3. ინდექსებიდან ამოვიღოთ პარამეტრი.

4. გავასწოროთ იტერაციები ისე, რომ ყველა იტერაციას ჰქონდეს ერთნაირი საზღვარი.

მიღებული რეგულარული სქემის გადაწყვეტა შესაძლებელია ტაბლო კალკულუსის მეშვეობით რომელსაც ეწოდება *STAB*. ამოხსნის სტრატეგია არის შემდეგი: თუ შესაძლებელია, პირველად გამოვიყენოთ პროპოზიციული კავშირების დაშლის წესები, შემდეგ გამოვიყენოთ ე.წ. ჩაციკლვის წესი ან დახურვის წესი. თუ ამ წესებიდან არც ერთის გამოყენება არ არის შესაძლებელი, მაშინ მაქსიმალურ ინტერვალზე არსებული იტერაცია უნდა დაიშალოს იტერაციის დაშლის წესებით.

გარდაქმნის სრულად აღწერისთვის და რეგულარული სქემების გადაწყვეტის სრული პროცედურის სანახავად დაინტერესებულ მკითხველებს გადავამისამართებთ [ACP11]. □

გადაწყვეტადობისათვის საკვანძო წერტილი არის ის, რომ ინდექსები შეიცავენ დაბმულ ცვლადებს კოეფიციენტებით $\leq 1$. თუ დავუშვებთ, რომ კოეფიციენტი შეიძლება იყოს ერთზე მეტი, მაშინ კლასი ხდება არაგადაწყვეტადი. უფრო გადაწყვეტადობისა და არაგადაწყვეტადობის შესახებ დეტალური ინფორმაციისათვის იხილეთ [ACP11].

## 2.2 სეკვენციათა კალკულუსი LKS

ამ ნაწილში ჩვენ განვსაზღვრავთ კლასიკურ პროპოზიციულ სეკვენციათა კალკულუსის LK-ის ვერსიას ფორმულათა სქემებისათვის, რომელიც განსხვავდება LK-სგან ორი რამით: კალკულუსის წესები მოქმედებენ რეგურალურ ფორმულათა სქემებზე, და დაშვებულია სპეციალური საწყისი სეკვენციები, რომლებსაც ეწოდებათ *მტკიცებათა ბმულები*. შევნიშნოთ, რომ [ACP09]-ში ტაბლო კალკულუსი *STAB* ფორმულათა სქემებისთვის უკვე განხილულია. ჩვენი კალკულუსი მისგან განსხვავდება რამდენიმე რამით: ყველაზე მნიშვნელოვანია, რომ დამტკიცებათა წარმოდგენის ფორმატი გაუმჯობესებულია და ადამიანის მიერ წაკითხვადია (STAB ფორმატი მხოლოდ მანქანისათვისაა განკუთვნილი და რეალიზებულია დამამტკიცებელში regSTAB). მეორე მნიშვნელოვანი ასკექტი ეხება ჩაციკლვის წესს, რომლის გამოყენების ნაცვლად, ჩვენ ვიყენებთ განსხვავებულ მიდგომას, რომელიც ეფუძნება მტკიცებების რეკურსიულ სპეციფიკაციებს, რომელიც უფრო შესაბამისია იმ დამტკიცებების ფორმალიზაციისათვის რომელიც აღმოჩენილია ადამიანის მიერ.

**განსაზღვრება 2.2.1** (სეკვენციის სქემა)**.** $\Gamma \vdash \Delta$ სახის გამოსახულებას, სადაც $\Gamma$ და $\Delta$ არიან სასრული (შესაძლებელია ცარიელი) ფორმულათა სქემების სიმრავლე, ეწოდებათ სეკვენციის სქემა. თუ $S$ არის სეკვენციის სქემა და $a$ არის არითმეტიკული გამოსახულება, მაშინ $S(a)$ განისაზღვრება ისევერ, როგორც ფორმულათა სქემების შემთხვევაში.

სეკვენციათა სქემების სემანტიკა განსაზღვრულია ანალოგიური გზით, როგორც სეკვენციებისათვის, მაგრამ იმ ინტერპრეტაციების

გათვალისწინებით, რომლებიც განსაზღვრულია პარაგრაფში 2.1.2. საწყისი სეკვენციის სქემა განსაზღვრულია შემდეგი ფორმით: $A \vdash A$, სადაც $A$ არის ნებისმიერი ატომური ფორმულათა სქემა. ამიერიდან, სეკვენციათა სქემებს მომავალში ჩვენ მოვიხსენიებთ, როგორც სეკვენციებს.

**განსაზღვრება 2.2.2** (მტკიცებათა ბმულები)**.** ვთქვათ, $\varphi$ არის მტკიცებათა სიმბოლო, $a$ არის არითმეტიკული გამოსახუალება, და $S$ სეკვენციის სქემა, მაშინ გამოსახულებას $\dfrac{(\varphi(a))}{S}$ ეწოდება *მტკიცებათა ბმული*.

ჩვენ ვნახეთ, რომ წრფივად-შემოსაზღვრული სქემების გამომსახველობითი უნარი არის რეგულალური სქმების მსგავსი. რადგანაც რეგულარული სქემები უფრო მარტივია და მატთან მუშაობა ადვილია, ამიტომ ამიერიდან ჩვენ განვიხილავთ მხოლოდ რეგულალურ სქემებს.

სქემატური LK-კალკულუსი შეიცავს ჩვეულებრივ კლასიკური პროპოზიციული სეკვენციათა LK კალკულუსის წესებს იმ პირობით, რომ ფორმულათა სქემები განიხილებიან ტოლობების $A(0) = \bigwedge_{i=0}^{0} A(i)$ და $(\bigwedge_{i=0}^{n} A(i)) \wedge A(n+1) = \bigwedge_{i=0}^{n+1} A(i)$ გათვალისწინებით (ანალოგიურად $\bigvee$ სიმბოლოსათვის)[1].

**განსაზღვრება 2.2.3** (LKS კალკულუსი)**.** სეკვენციათა კალკულუსი LKS შეიცავს საწყის სეკვენციათა სქემებს ან მტკიცებათა ბმულებს, როგორც აქსიომები და შედგება პროპოზიციული და სტრუქტურული LK გამოყვანის წესებისაგან.

**განსაზღვრება 2.2.4** (LKS-დამტკიცება და LKS-ქვედამტკიცება)**.** LKS-დამტკიცება არის ხე, რომლის წვეროებზეც არის საწყისი სეკვენციები ან დამტკიცებათა ბმულები, სხვა კვანძებზე არის სეკვენციები და ტოტები არის გამოყვანის წესები. ხის ფუძეში არსებულ სეკვენცია $S$-ს ეწოდება საბოლოო სეკვენცია და იწერება ბოლოში. თუ $\phi$ არის $\Gamma \vdash \Delta$ საბოლოო სეკვენციის LKS-დამტკიცება, მაშინ შეგვიძლია დავწეროთ:

$$\begin{array}{c} \phi \\ \Gamma \vdash \Delta \end{array}$$

[1]ეს ტოლობები გამოიყენება იმგვარად, რომ $\wedge$ და $\vee$ ლოგიკური კავშირების წესები გამოყენებული იქნას $\bigwedge$ და $\bigvee$ ოპერატორებისათვის.

და ვამბობთ, რომ $\Gamma \vdash \Delta$ არის დამტკიცებადი LKS-ში. LKS-დამტკიცებას განკვეთის წესის გარეშე ეწოდება *განკვეთის წესისაგან თავისუფალი*. $\phi$-ს LKS-ქვემტკიცება არის $\phi$-ს ნებისმიერი ქვე-ხე, რომელიც აგრეთვე არის LK-დამტკიცება.

ფორმულათა სქემების ანალოგიურად, ჩვენ გამოვიყენებთ აღნიშვნებს $\pi(k)$ და $\pi(a)$. LKS-დამტკიცებას ეწოდება *შემოსაზღვრული*, თუ ის არ შეიცავს ინდექსის ცვლადებს და მტკიცებათა ბმულებს. შევნიშნოთ, რომ შემოსაზღვრული LKS-დამტკიცება რეალურად არის LK-დამტკიცება (რომელიც მიღებულია შემოსაზღვრული $\bigwedge, \bigvee$ ოპერატორების ჩანაცვლებით, შესაბამისად, სასრული რაოდენობა $\wedge, \vee$ ოპერატორებით).

პრაქტიკული გამოყენებისას გვჭირდება რეკურსიის ცნების დამატება, იმისთვის რომ LKS-დამტკიცებები განსაზღვრული იყოს ისე, როგორც ზემოთ გვაქვს განხილული. შესაბამისად მივიღებთ *მტკიცებათა სქემების* ცნებას:

**განსაზღვრება 2.2.5** (მტკიცებათა სქემები)**.** ვთქვათ $\psi_1, \dots, \psi_\alpha$ არის დამტკიცებათა სიმბოლოები და $S_1(n), \dots, S_\alpha(n)$ კი მათი შესაბამისი საბოლოო სეკვენციები. მაშინ *მტკიცებათა სქემა* $\Psi$ არის წყვილების სიმრავლე[2]

$$\langle (\pi_1, \nu_1(k)), \dots, (\pi_\alpha, \nu_\alpha(k)) \rangle$$

ისეთი, რომ:

1. ყოველი წყვილი $(\pi_\beta, \nu_\beta(k))$ დაკავშირებულია $\psi_\beta$ სიმბოლოსთანსთან, ყოველი $\beta = 1, \dots, \alpha$-სთვის,

2. $\pi_\beta$ არის $S_\beta(0)$ სეკვენციის შემოსაზღვრული LKS-დამტკიცება, ყოველი $\beta = 1, \dots, \alpha$-სათვის,

3. $\nu_\beta(k)$ არის $S_\beta(k+1)$ სეკვენციის LKS-დამტკიცება, ისეთი, რომ $\nu_\beta(k)$ შეიცავს მხოლოდ ერთ $k$ პარამეტრს და შემდეგი ფორმის მტკიცებათა ბმულებს:

$$\frac{(\psi_\beta(k))}{S_\beta(k)} \quad \text{და/ან} \quad \frac{(\psi_\gamma(a))}{S_\gamma(a)}$$

[2] ზოგჯერ შესაძლოა დავწეროთ $\langle \psi_1, \dots, \psi_\alpha \rangle$

სადაც $\beta < \gamma$ და $a$ არის ზოგადი არითმეტიკული გამოსახულება.

ჩვენ ვგულისხმობთ $\pi_\beta$ და $\nu_\beta(k)$ დამტკიცებების საბოლოო სეკვენციებში ფორმულათა შემოსვლის იდენტიფიცირების შესაძლებლობას, რომ ვილაპარაკოთ $\psi_\beta$-სა საბოლოო სეკვენციაში ფორმულათა შემოსვლაზე. ჩვენ ასევე ვიტყვით, რომ $S_1(n)$ არის $\Psi$ დამტკიცებათა სქემის საბოლოო სეკვენცია.

ეხლა განვიხილოთ მტკიცებათა სქემების სინტაქსური მნიშვნელობა: მტკიცებათა სქემა $\Psi$ საბოლოო სეკვენციით $S(n)$ შესაძლებელია განხილულ იქნას როგორც შემოსაზღვრული LKS-დამტკიცებების მიმდევრობა $(\pi_0, \pi_1, ...)$ რომლებიც ამტკიცებენ შესაბამისად სეკვენციებს $S(0), S(1), ....$ აღნიშნული განსაზღვრების ფორმალიზაციისათვის ჩვენ LKS-დამტკიცებებს განვიხილავთ როგორც ტერმებს და განვსაზღვრავთ მათზე გადაწერის სისტემას. ამ სისტემის ნორმალური ფორმა $\alpha \in \mathbb{N}$ უნდა იყოს ზუსტად $\pi_\alpha$ დამტკიცება ზემოთ მოყვანილი მიმდევრობიდან.

**განსაზღვრება 2.2.6** (მტკიცებათა სქემების შეფასება)**.** ვთქვათ, $\Psi$ არის მტკიცებათა სქემა განსაზღვრული 2.2.5 განსაზღვრების შესაბამისად. ჩვენ განვსაზღვრავთ გადაწერის წესებს მტკიცებათა ბმულებისთვის

$$\frac{(\psi_\beta(0))}{\bar{S}} \to \pi_\beta, \quad \text{და} \quad \frac{(\psi_\beta(k+1))}{\bar{S}} \to \nu_\beta(k)$$

ყოველი $1 \leq \beta \leq \alpha$.

ახლა, ყოველი $\gamma \in \mathbb{N}$ ჩვენ განვსაზღვრავთ $\psi_\beta \downarrow_\gamma$ როგორც შემდეგ ნორმალურ ფორმას $\frac{(\psi_\beta(\gamma))}{\bar{S}(\gamma)}$ ზემოთ მოყვანილი გადაწერის სისტემის გათვალისწინებით. უფრო მეტიც, ჩვენ განვსაზღვრავთ $\Psi\downarrow_\gamma = \psi_1\downarrow_\gamma$.

**შენიშვნა 2.2.7.** უფრო კონკრეტულად, $\psi_\beta \downarrow_\gamma$ არის შემდეგის ნორმალური ფორმა $\frac{(\psi_\beta(\bar{\gamma}))}{\bar{S}(\bar{\gamma})}$ სადაც $\bar{\gamma}$ რიცხობრივად შესაბამისია $\gamma$-სი.

**მაგალითი 2.2.8.** წარმოგიდგენთ შემდეგ მტკიცებათა სქემებს $\Psi$:

$$\langle(\pi, \nu(k))\rangle,$$

საადაც $\pi$ არის:

$$\frac{p_0 \vdash p_0 \quad p_1 \vdash p_1}{p_0, p_0 \Rightarrow p_1 \vdash p_1} \Rightarrow : l$$

და $\nu(k)$:

$$\dfrac{\dfrac{\overset{(\psi(k))}{\overline{\qquad\qquad\qquad\qquad}}}{p_0, \bigwedge_{i=0}^{k}(p_i \Rightarrow p_{i+1}) \vdash p_{k+1} \quad p_{k+2} \vdash p_{k+2}} \Rightarrow : l}{p_0, \bigwedge_{i=0}^{k}(p_i \Rightarrow p_{i+1}), p_{k+1} \Rightarrow p_{k+2} \vdash p_{k+2}} \wedge : l}{p_0, \bigwedge_{i=0}^{k+1}(p_i \Rightarrow p_{i+1}) \vdash p_{k+2}}$$

მაშინ, განსაზღვრება 2.2.6-ის შესაბამისად, $\Psi \downarrow_0$ არის მხოლოდ $\pi$; $\Psi \downarrow_1$ არის:

$$\dfrac{\dfrac{\dfrac{p_0 \vdash p_0 \quad p_1 \vdash p_1}{p_0, p_0 \Rightarrow p_1 \vdash p_1} \Rightarrow : l \qquad p_2 \vdash p_2}{p_0, p_0 \Rightarrow p_1, p_1 \Rightarrow p_2 \vdash p_2} \Rightarrow : l}{p_0, (p_0 \Rightarrow p_1) \wedge (p_1 \Rightarrow p_2) \vdash p_2} \wedge : l$$

$\Psi \downarrow_2$ არის:

$$\dfrac{\dfrac{\dfrac{\dfrac{p_0 \vdash p_0 \quad p_1 \vdash p_1}{p_0, p_0 \Rightarrow p_1 \vdash p_1} \Rightarrow : l \qquad p_2 \vdash p_2}{p_0, p_0 \Rightarrow p_1, p_1 \Rightarrow p_2 \vdash p_2} \Rightarrow : l}{p_0, (p_0 \Rightarrow p_1) \wedge (p_1 \Rightarrow p_2) \vdash p_2} \wedge : l \qquad p_3 \vdash p_3}{p_0, (p_0 \Rightarrow p_1) \wedge (p_1 \Rightarrow p_2), p_2 \Rightarrow p_3 \vdash p_3} \Rightarrow : l}{p_0, (p_0 \Rightarrow p_1) \wedge (p_1 \Rightarrow p_2) \wedge (p_2 \Rightarrow p_3) \vdash p_3} \wedge : l$$

და ა.შ.

ახლა, ვაჩვენოთ, რომ მტკიცებათა სქემების შეფასების განსაზღვრება ნამდვილად სწორია.

**წინადადება 2.2.1** (სისწორე)**.** *ვთქვათ* $\Psi : \langle(\pi_1, \nu_1(k)), \ldots, (\pi_\alpha, \nu_\alpha(k))\rangle$ *არის მტკიცებათა სქემა საბოლოო სეკვენციით* $S(n)$*. მაშინ, ყოველი* $\gamma \in \mathbb{N}$ *და* $1 \leq \beta \leq \alpha$*,* $\psi_\beta \downarrow_\gamma$ *არის შემოსაზღვრული* LKS*-დამტკიცება საბოლოო სეკვენციით* $S_\beta(\gamma)$*. მაშასადამე,* $\Psi \downarrow_\gamma$ *არის შემოსაზღვრული* LKS*-დამტკიცება საბოლოო სეკვენციით* $S(\gamma)$*.*

*დამტკიცება.* საჭიროა ორმაგი ინდუქცია $\alpha$ და $\gamma$ პარამეტრების მიმართ. დავუშვათ $\alpha = 1$. თუ $\gamma = 0$, მაშინ დამტკიცების ბმული გადაიწერება $\pi_1$-ში, რაც გვინდოდა განსაზღვრების თანახმად. ეხლა დავუშვათ, რომ $\gamma > 0$ და წინადადება სრულდება ყოველი ნატურალური რიცხვისათვის, რომელიც ნაკლებია $\gamma$-ზე. განსაზღვრების თანახმად, $\nu_1(k)$ შესაძლოა

შეიცავდეს დამტკიცებათა ბმულს მხოლოდ $\psi_1(k)$-ზე და ინდუქციის დაშვების თანახმად $\frac{(\psi_1(\gamma-1))}{S_1(\gamma-1)}$ გადაიწერება $S_1(\gamma-1)$ სეკვენციის LKS-დამტკიცებაში. მაშასადამე, დამტკიცების ბმული $\psi_1(\gamma)$-ზე გადაიწერება $S_1(\gamma)$ სეკვენციის LKS-დამტკიცებაში. მიღებული ყველა LKS-დამტკიცება შემოსაზღვრულია, ვინაიდან $\pi_1$ შემოსაზღვრულია და $k$ არის ერთადერთი პარამეტრი, რომელსაც შეიცავს $\nu_1(k)$.

ეხლა, დავუშვათ, რომ $\alpha > 1$ და წინადადება სრულდება ყველა მტკიცებათა სქემისათვის, რომელიც შეიცავს $\alpha$-ზე ნაკლებ დამტკიცებათა სიმბოლოს. ისევ, თუ $\gamma = 0$, მაშინ დამტკიცების ბმული გადაიწრება $\pi_\beta$-ში, ყოველი $1 \leq \beta \leq \alpha$. თუ $\gamma > 0$, მაშინ ყოველი დამტკიცების ბმულისათვის $\psi_\beta(\gamma-1)$-ზე, ზემოთ მოყვანილი არგუმენტები გამოიყენება. ამიტომ, განვიხილოთ დამტკიცებათა ბმულები $\psi_{\beta'}(\gamma')$-ზე, სადაც $\beta' > \beta$. მაშინ ცხადია, დამტკიცებატა სქემას $\Psi'$: $\langle(\pi_{\beta'}, \nu_{\beta'}(k)), \ldots, (\pi_\alpha, \nu_\alpha(k))\rangle$ აქვს უფრო ნაკლები დამტკიცების სიმბოლო, ვიდრე $\Psi$-ს, შესაბამისად შეგვიძლია გამოვიყენოთ (გარე) ინდუქციის დაშვება, რომ წინადადება სრულდება $\Psi'$-სათვის. მაშინ, ცხადია, $\psi_\beta\downarrow_\gamma$ არის $S_\beta(\gamma)$ სეკვენციის შემოსაზღვრული LKS-დამტკიცება ყოველი $1 \leq \beta \leq \alpha$. □

შევნიშნოთ, რომ LKS კალკულუსი საზოგადოდ არ არის სრული, მაგრამ ის არის სრული რეგულარული სქემების ქვესიმრავლეზე, სადაც ყველა იტერაცია დალაგებულია $[0, n+\alpha]$ ინტერვალზე ნებისმიერი $\alpha \in \mathbb{N}$. ასეთი ზოგადმართებული რეგულარული სქემისათვის, მდკიცებათა სქემის პოვნა შესაძლებელია წინადადება 2.1.2-ის დამტკიცებაში მოცემული მეთოდის მსგავსად. განსხვავება არის ის, რომ ჩაციკლვის წესის ნაცვლად ჩვენ გამოვიყენებთ დამტკიცების ბმულს.

## 2.3 პირველი რიგის მტკიცებათა სქემები

ჩვენ განვსაზღვრავთ პირველი რიგის სქემატურ ენას [DLRW15]-ის მიხედვით, რომელიც არის [ACP09, ACP11]-ში აღწერილი ენის გაფართოება

პირველი რიგის ლოგიკისათვის. ეს საშუალებას მოგვცემს პირველი რიგის ფორმულათა უსასრულო მიმდევრობა წარმოვადგინოთ სასრული სახით.

ჩვენ განვიხილავთ ორი სახეობის ცვლადებს: $\omega$, რომ გამოვსახოთ ნატურალური რიცხვები, და $\iota$, რომ გამოვსახოთ პირველი რიგის დომენი. ჩვენი ენა შედგება აღნიშნული სახეობის ცვლადების თვლადი სიმრავლეებისაგან, $n$-ური ფუნქციონალური და პრედიკატული სიმბოლოებისაგან, რომლებიც დაყოფილია მუდმივ ფუნქციონალურ/პრედიკატულ და განსასაზღვრ ფუნქციონალურ/პრედიკატულ სიმბოლოებად. პირველი გამოიყენება ჩვეულებრივი პირველი რიგის ტერმებისა და პრედიკატების განსასაზღვრად, ხოლო მეორე საშუალებას გვაძლევს ვაწარმოოთ ამ ტერმებსა და ფორმულებზე რეკურსია.

ტერმები აგებულია ცვლადებისა და მუდმივი ფუნქციონალური სიმბოლოებისაგან სტანდარტული ინდუქციური გზით. ნატურალური რიცხვები განისაზღვრება სტანდარტული გზით ფუნქციების $0 : \omega$ და $s : \omega \to \omega$ გამოყენებით. $V(t)$ აღნიშნავს $t$ თერმის ცვლადების სიმრავლეს, და $\bar{}$ ჩვენ აღვნიშნავთ ტერმთა მიმდევრობებს.

ყოველი განსასაზღვრი ფუნქიონალური სიმბოლოსათვის $f$, ჩვენ ვგულისხმობთ, რომ მისი ტიპი არის $\omega \times \tau_1 \times \cdots \times \tau_n \to \tau$ (სადაც $n \geq 0$ და $\tau ::= \omega \mid \iota \mid \tau \to \tau$). მაშინ ჩვენ განვსაზღვრავთ ტერმთა სქემას $f(y, \overline{x})$ შემდეგი ორი გადაწერის წესით:

$$f(0, \overline{x}) \to t_0 \qquad f(s(y), \overline{x}) \to t[f(y, \overline{x})]$$

სადაც $V(t_0) \subseteq \{x_1, \ldots, x_n\}$ და $V(t[f(y, \overline{x})]) \subseteq \{y, x_1, \ldots, x_n\}$, და $t_0, t$ არიან ტერმები, რომლებშიც არ გვხვდება $f$. თუ განსასაზღვრი ფუნქციონალური სიმბოლო $g$ გვხვდება $t_0$ ან $t$-ში, მაშინ ვიტყვით, რომ $g \prec f$. ჩვენ ვგულისხმობთ, რომ გადაწერის წესები პრიმიტულ რეკურსულია, რაც ნიშნავს იმას, რომ მოითხოვება $\prec$ დალაგების არარეფლექსურობა.

ჩვენ ჩავწერთ $t \twoheadrightarrow t'$ იმ ფაქტის აღსანიშნად, რომ გამოსახულება $t$ გადაიწერება გამოსახულება $t'$-ში (ნებისმიერი რაოდენობა საფეხურით).

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}\neg_{\mathsf{l}} \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\neg_{\mathsf{r}}$$

$$\frac{\Gamma \vdash \Delta, A \quad B, \Gamma \vdash \Delta}{A \Rightarrow B, \Gamma \vdash \Delta}\Rightarrow_{\mathsf{l}} \qquad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B}\Rightarrow_{\mathsf{r}}$$

$$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\wedge_{\mathsf{l}} \qquad \frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}\wedge_{\mathsf{r}}$$

$$\frac{A, \Gamma \vdash \Delta \quad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta}\vee_{\mathsf{l}} \qquad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}\vee_{\mathsf{r}}$$

$$\frac{A(t), \Gamma \vdash \Delta}{\forall x A(x), \Gamma \vdash \Delta}\forall_{\mathsf{l}} \qquad \frac{\Gamma \vdash \Delta, A(u)}{\Gamma \vdash \Delta, \forall x A(x)}\forall_{\mathsf{r}}{}^{1}$$

$$\frac{A(u), \Gamma \vdash \Delta}{\exists x A(x), \Gamma \vdash \Delta}\exists_{\mathsf{l}}{}^{1} \qquad \frac{\Gamma \vdash \Delta, A(t)}{\Gamma \vdash \Delta, \exists x A(x)}\exists_{\mathsf{r}}$$

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\mathsf{c}_{\mathsf{l}} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\mathsf{c}_{\mathsf{r}} \qquad \frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\mathsf{w}_{\mathsf{l}} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\mathsf{w}_{\mathsf{r}}$$

$$\frac{}{a \vdash a}\mathsf{ax} \qquad \frac{\varphi(k, \vec{t})}{S(k, \vec{t})}\,\mathsf{pax}^{2} \qquad \frac{S[t]}{S[t']}\,\mathcal{E}^{3}$$

[1] $u$ არის შესაბამისი ტიპის უნიკალური ცვლადი, რომელიც არ გვხვდება $\Gamma$, $\Delta$ და $A(x)$-ში.

[2] $\varphi$ არის დამტკიცების სიმბოლო და $S$ სეკვენცია, სადაც თავისუფალი ცვლადები ჩანაცვლებულია $\vec{t}$ თერმებით.

[3] მოცემულია ტოლობების სასრული სიმრავლე $\mathcal{E}$, სადაც $\mathcal{E} \models t = t'$.

ფიგურა 2.1: პირველი რიგის სეკვენციათა კალკულუსი LKS.

**მაგალითი 2.3.1.** დავუშვათ, $g\colon \omega \times \iota \to \iota$ არის განსასაზღვრი ფუნქციონალური სიმბოლო, $f\colon \iota \to \iota$ არის მუდმივი ფუნქციონალური სიმბოლო, და $k\colon \omega, x\colon \iota$ არიან ცვლადები. შემდეგი გადაწერის წესები $g$-სათვის

$$g(0, x) \to x \qquad g(s(k), x) \to f(g(k, x))$$

განსაზღვრავენ ტერმთა სქემას $g(k, x)$. ყოველი ნატურალური რიცხვისათვის $n$, $g(n, x)$ გადაიწერება $f^n(x)$, მაგალითად,

$$g(s(s(s(0))), x) \twoheadrightarrow f(f(f(x))).$$

ფორმულები აიგება სტანდარტული ინდუქციური გზით ატომალური ფორმულებისაგან ლოგიკური ოპერატორების $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$ და $\exists$ გამოყენებით. ინტერპრეტაციის, შესრულებადობის და მართებულობის ცნებები განსაზღვრულია სტანდარტული კლასიკური აზრით.

განსასაზღვრი ფუნქიონალური სიმბოლოების მსგავსად, გადაწერის წესები მოცემულია განსასაზღვრი პრედიკატული სიმბოლობისთვისაც, რაც გვაძლევს ფორმულათა სქემებს იმ პირობით, რომ განსასაზღვრ პრედიკატულ სიმბოლოებსაც გააჩნიათ არარეფლექსური დალაგება $\prec$. შემდგომში ფორმულათა სქემებს შემოკლებიტ ფორმულებად მოვიხსენიებთ.

ფორმულაში ცვლადის შემოსვლას ეწოდება დაბმული, თუ ეს ცვლადი არის $\forall$ ან $\exists$ ლოგიკური ოპერატორების მოქმედების არეში, წინააღმდეგ შემთხვევაში მათ ეწოდებათ თავისუფალი ცვლადები. ჩვენ შემთხვევაში ძალზედ მნიშვნელოვანია ერთიდაიმავე დაბმული ცვლადის სხვადასხვა შემოსვლის სწორი ინტერპრეტაცია: $x$ დაბმული ცვლადის შემოსვლა ასოცირდება მასთან ყველაზე ახლოს მდგომ კვანტორთან, რომელიც მას აბამს. შემდეგი მაგალითი ახდენს ამ ინტერპრეტაციის ილუსტრაციას.

**მაგალითი 2.3.2.** დავუშვათ, $P\colon \omega$ არის განსასაზღვრი პრედიკატული სიმბოლო, $Q\colon \omega \times \iota$ არის მუდმივი პრედიკატული სიმბოლო, და $k\colon \omega, x\colon \iota$ ცვლადები. შემდეგი გადაწერის წესები $P$-სათვის

$$P(0) \to \forall x Q(0,x) \qquad P(s(k)) \to \exists x(Q(k,x) \wedge P(k)).$$

განსაზღვრავენ ფორმულათა სქემას $P(k)$. მაშინ, მაგალითად,

$$P(s(s(0))) \twoheadrightarrow \exists x(Q(s(s(0)),x) \wedge \exists x(Q(s(0),x) \wedge \forall x Q(0,x)))$$

რაც ექვივალენტურია (დაბმული ცვლადების სახელის გადარქმევით)

$$\exists x_2(Q(s(s(0)),x_2) \wedge \exists x_1(Q(s(0),x_1) \wedge \forall x_0 Q(0,x_0))).$$

**წინადადება 2.3.1.** *ვთქვათ $A$ არის ფორმულა, მაშინ $A$-დან დაწყებული ნებისმიერი გადაწერა დასრულებადია და $A$-ს გააჩნია ერთადერთი ნორმალური ფორმა.*

*დამტკიცება.* ტრივიალურია, რადგან ყველა განსაზღვრება არის დაფუძნებული პრიმიტიულ რეკურსიაზე. □

სეკვენცია არის შემდეგი ფორმის გამოსახულება: $\Gamma \vdash \Delta$, სადაც $\Gamma$ და $\Delta$ არიან ფორმულათა სქემების მულტი სიმრავლეები. სეკვენციები აღინიშნება $S(\overline{x})$, სადაც $\overline{x}$ აღნიშნავს თავისუფალ ცვლადებს $S$-ში.

სეკვენციათა კალკულუსი LKS მოცემულია ფიგურა 2.1-ში, სადაც მტკიცებათა აქსიომები (pax), რომელსაც უწოდებენ მტკიცებათა დამაკავშირებელს [DLRW15, Ruk13]-ში, შეიძლება შეგვხვდეს მხოლოდ დამტკიცების ტოტების სათავეში. მტკიცებათა აქსიომას აქვს იგივე დატვირთვა, რაც ინდუქციის ჰიპოთეზას: დაშვებულია, რომ დამტკიცება $\varphi$ საფეხურზე $k$ ამტკიცებს სეკვენციას $S(k)$.

ამ უკანასკნელს მივყავართ მტკიცებათა სქემების განსაზღვრებამდე: $\varphi_1, \dots, \varphi_n$ სიმბოლოებით აღნიშნული LKS დამტკიცებათა წყვილებისაგან შემდგარი მრავალწევრი $\Psi$ არის დამტკიცებათა სქემა, სადაც თითოეული წყვილი შედგება ინდუქციური განსაზღვრების საბაზისო და რეკურსული შემთხვევებისაგან. დამტკიცებათა სიმბოლობი უნდა იყოს სწორად დალაგაბული შემდეგი აზრით: თუ $i > j$, მაშინ $\varphi_i$ არ უნდა შეიცავდეს მტკიცებატა აქსიომას, რომელიც მიუთითებს $\varphi_j$. ასევე ვიტყვით, რომ $\varphi_1$-ს საბოლოო სეკვენცია არის $\Psi$ მტკიცებატა სქემის საბოლოო სეკვენცია. მტკიცებათა სქემების ფორმალური განსაზღვრებისათვის დაინტერესებულმა მკითხველმა იხილეთ [DLRW15, Ruk13].

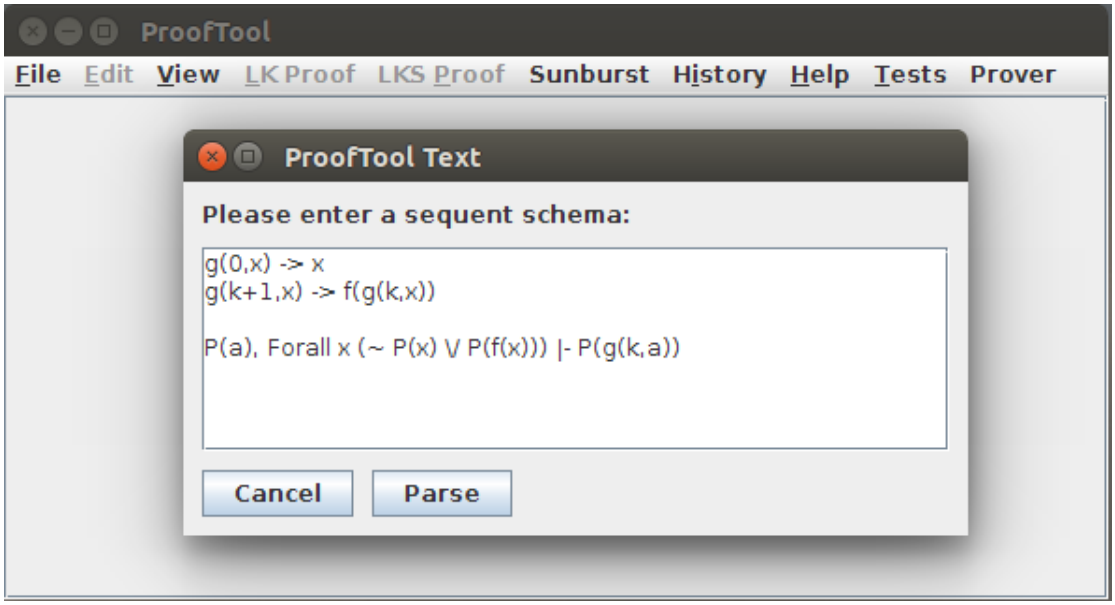
**მაგალითი 2.3.3.** ვთქვათ, $g(k, x)$ არის თერმთა სქემა, რომელიც განსაზღვრულია მაგალითი 2.3.1-ში. მაშინ, სეკვენციის $P(a), \forall x(\neg P(x) \vee P(f(x))) \vdash P(g(k, x))$ დამტკიცებათა სქემა მოცემულია ფიგურა 2.3 (საბაზისო შემთხვევა) და ფიგურა 2.4-ში (რეკურსული შემთხვევა) [3].

ზემოთ მოცემული განსაზღვრებების შედეგად ადვილი დასანახია, რომ მტკიცებათა სქემა ბუნებრივად წარმოადგენს პირველი რიგის დამტკიცებათა უსასრულო მიმდევრობას.

**წინადადება 2.3.2.** *სეკვენციათა კალკულუსი* LKS *არის კორექტული.*

*დამტკიცება.* მიუხედავად იმისა, რომ კალკულუსი იგივე არ არის, დამტკიცება [DLRW15, Ruk13]-ში არსებული ანალოგიური შედეგის დამტკიცების მსგავსია. □

[3]ეს დამტკიცებათა სქემა მიღებულია ავტომატური დამამტკიცებლის მიერ, ამიტომაც შეიცავს ზედმეტ სტრუქტურულ წესებს. დეტალები იხილეთ რეალიზაციის სექციაში.



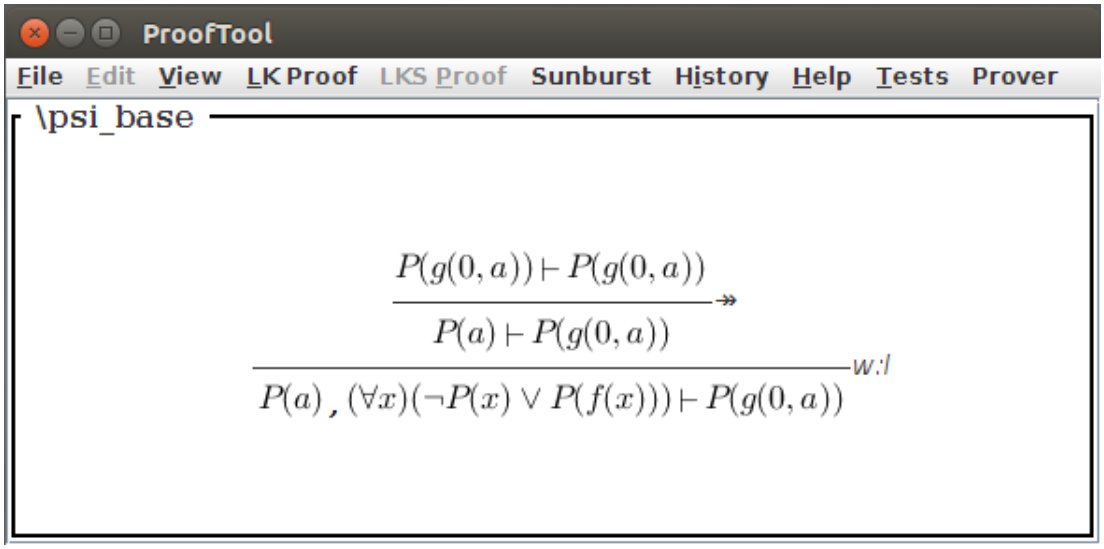

ფიგურა 2.2: დავალების მაგალითი ალგორითმისათვის.

## 2.4 მტკიცებათა სქემების აგების ალგორითმი

მოცემული სეკვენციისათვის დამტკიცებათა სქემის აგება იყოფა ორ ნაწილად: ჯერ ავაგოთ LKS-დამტკიცების საბაზისო ნაწილი და შემდგომ LKS-დამტკიცების რეკურსიული ნაწილი. მთავარი განსხვავება ამ ორ პროცედურას შორის არის დამტკიცებათა აქსიომების გამოყენება, ე.წ. ინდუქციის დაშვებები, რომელიც საბაზო ნაწილში სრულიად ზედმეტია.

კვანტორებთან სამუშაოდ ვიყენებთ [AMS98]-ში აღწერილი მეთოდის ანალოგს. ეს ნიშნავს, რომ სუსტი კვანტორებისათვის ჩანაცვლების თერმების არჩევა გადადებულია, სანამ იგი არ მიიღება უნიფიკაციის საშუალებით. იგი მუშაობს შემდეგნაირად: სუსტი კვანტორული ფორმულის[4] დაშლისას ჩვენ აგრეთვე ვიტოვებთ მის ორიგინალ ვერსიას (რათა თავიდან ავიცილოთ უკან დაბრუნება სხვა თერმის შესარჩევად) და ვანაცვლებთ კვანტორის ცვლადს ახალი უნიკალური ცვლადით, რომელიც არ გვხვდება დამტკიცებაში. ეს ცვლადი მიჰყვება ფორმულებს აქსიომებამდე, მანამ სანამ აქსიომების გამოყენებისას უნიფიკაციის მეთოდით არ მივიღებთ სწორ თერმს მის

[4]სუსტი კვანტორული ფორმულა არის ფორმულა, რომლის უკიდურესი ლოგიკური ოპერატორი არის სუსტი კვანტორი.

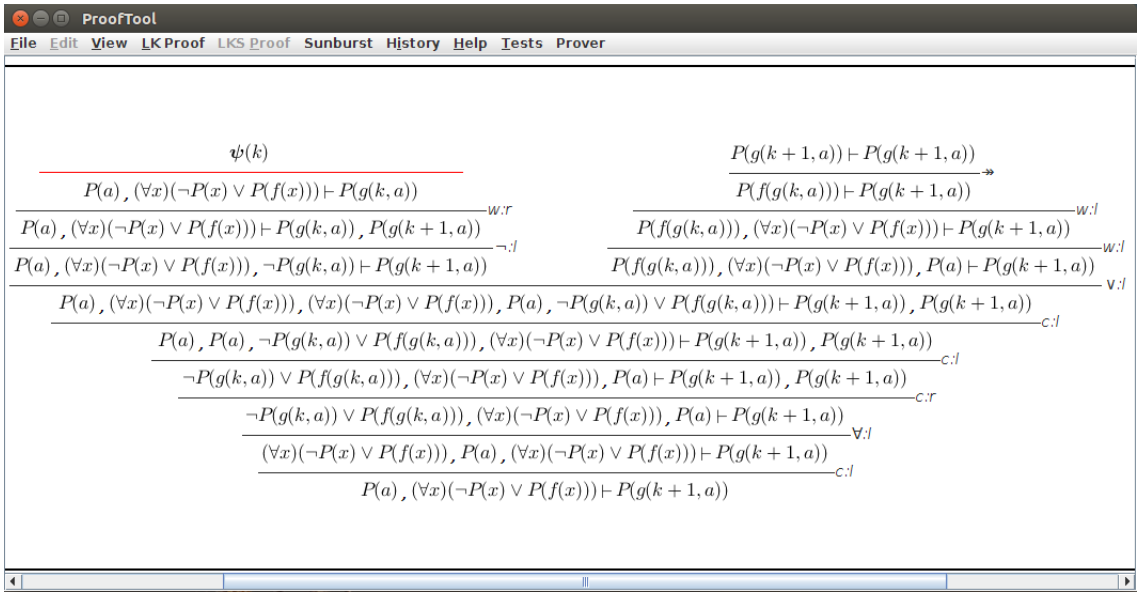



ფიგურა 2.3: LKS-დამტკიცება საბაზისო შემთხვევისათვის.

ადგილას. სწორი თერმის მიღების შემდეგ ჩანაცვლება ხორციელდება მთელს დამტკიცებაში, ყველგან სადაც გვხვდებოდა აღნიშნული ცვლადი.

საზოგადოდ, წესების გამოყენება არის არადეტერმინისტული, მაგრამ უკეთესი ეფექტის მისაღებად საჭიროა განვსაზღვროთ მათი რიგითობა. პრიორიტეტი ენიჭებათ პროპოზიციულ და ექვივალენტობის წესებს. აღსანიშნავია, რომ LKS-ში არსებული ყველა პროპოზიციული წესი არის ინვერსული, შესაბამისად ისინი შეიძლება იყოს გამოყენებული თავისუფლად. პრიორიტეტი ენიჭება უნარულ წესებს, რადგანაც ბინარული წესები ახდენენ სეკვენციის ფორმულების დუბლირებას. აქედან გამომდინარე, უნარული წესები გამოიყენება როდესაც შესაძლებელია და ბინარული წესების გამოყენება გადაიდება შეძლებისდაგვარად შორს.

რეკურსული დამტკიცების ძებნის დროს, განსაზღვრებადი ფუნქციონალური და პრედიკატული სიმბოლოები უნდა გადაიწეროს ისე, რომ თითოეული სიმბოლო გადაიწეროს მხოლოდ ერთი ნაბიჯით ქვემოთ (მაგ: $k+1$-დან $k$-მდე). გადაწერის შემდეგ, თუ აღარ არის სხვა წესები, რომელთა გამოიყენებაც შესაძლებელია ამ სეკვენციაზე, მაშინ შემოგვაქვს მტკიცებათა აქსიომა ან ალგორითმი უნდა დასრულდეს სიგნალით "დამტკიცება ვერ მოიძებნა". თუ სეკვენცია შეიცავს დასამტკიცებელი სეკვენციის შეთანადებულ შემთხვევას k პარამეტრის მიმართ, მაშინ მტკიცებათა აქსიომა




ფიგურა 2.4: LKS-დამტკიცება რეკურსული შემთხვევისათვის.

უნდა იყოს მიმართული თავის თავზე. სხვა შემთხვევაში, მტკიცებათა აქსიომა ახალი დამტკიცების სიმბოლოთი უნდა შეიქმნას და ამ სიმბოლოსა და სეკვენციისათვის ახალი დამტკიცების ძებნის პროცედურა დაიგეგმოს.

**მაგალითი 2.4.1.** ალგორითმის საილუსტრაციოდ განვიხილოთ სეკვენცია $S(n)\colon P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(g(n,a))$, სადაც $g(k,x)$ არის თერმთა სქემა, რომელიც განსაზღვრულია მაგალითი 2.3.1-ში, და ავაგოთ მისი დამტკიცების სქემა. საბაზისო შემთხვევაში, ალგორითმის მიხედვით, ვინაიდან არ გვაქვს გამოსაყენებელი პროპოზიციული წესები, პირველი გამოიყენებს ექვივალენტობის წესს და მივიღებთ $\varphi(0)$:

$$\dfrac{\dfrac{P(a) \vdash P(a)}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(a)}\,\mathsf{w_l}}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(g(0,a))}\,\mathcal{E}$$

რეკურსულ შემთხვევაშიც, ანალოგიურად დაიწყებს ექვივალენტობის წესით და შემდეგ განაგრძობს კვანტორული ფორმულის დაშლით. შესაბამისად მივიღებთ შემდეგ გამოყვანას:

$$\dfrac{\dfrac{\dfrac{\dfrac{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(u), \Delta \qquad \Gamma, P(f(u)) \vdash P(f(g(n,a)))}{P(a), \forall x(P(x) \Rightarrow P(f(x))), P(u) \Rightarrow P(f(u)) \vdash P(f(g(n,a)))}\Rightarrow_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(f(g(n,a)))}\forall_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(f(g(n,a)))}\mathsf{c}_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(g(s(n),a))}\mathcal{E}$$

ამ ეტაპზე, მარჯვენა მხარეს ალგორითმი გააკეთებს $P(f(u))$ და $P(f(g(n,a)))$ ატომების უნიფიკაციას, რათა დაასრულოს ტოტი აქსიომით. გვექნება უნიფიკატორი $\sigma = [u \mapsto g(n,a)]$ და მივიღებთ გამოყვანას:

$$\dfrac{\dfrac{\dfrac{\dfrac{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(g(n,a)), \Delta \quad \dfrac{P(f(g(n,a))) \vdash P(f(g(n,a)))}{\Gamma, P(f(g(n,a))) \vdash P(f(g(n,a)))}\mathsf{w}_{\mathsf{l}} \times 2}{P(a), \forall x(P(x) \Rightarrow P(f(x))), P(g(n,a)) \Rightarrow P(f(g(n,a))) \vdash P(f(g(n,a)))}\Rightarrow_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(f(g(n,a)))}\forall_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(f(g(n,a)))}\mathsf{c}_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(g(s(n),a))}\mathcal{E}$$

მარცხენა მხარეს, ალგორითმი აღმოაჩენს, რომ მიიღო იგივე სეკვენცია, რომელიც შეიცავს $S(n)$-ს, შესაბამისად დასვამს დამტკიცების აქსიომას და საბოლოოდ მივიღებთ დამტკიცებას $\varphi(n+1)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\varphi(n)}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(g(n,a))}\mathsf{pax}}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(g(n,a)), \Delta}\mathsf{w}_{\mathsf{r}} \quad \dfrac{P(f(g(n,a))) \vdash P(f(g(n,a)))}{\Gamma, P(f(g(n,a))) \vdash P(f(g(n,a)))}\mathsf{w}_{\mathsf{l}} \times 2}{P(a), \forall x(P(x) \Rightarrow P(f(x))), P(g(n,a)) \Rightarrow P(f(g(n,a))) \vdash P(f(g(n,a)))}\Rightarrow_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(f(g(n,a)))}\forall_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(f(g(n,a)))}\mathsf{c}_{\mathsf{l}}}{P(a), \forall x(P(x) \Rightarrow P(f(x))) \vdash P(g(s(n),a))}\mathcal{E}$$

ადვილი დასანახია, რომ მოცემული ალგორითმი არ არის სრული. ფაქტობრივად, არ არსებობს სრული ალგორითმი, რადგან ფორმულათა სქემებისათვის შესრულებადობის პრობლემა არის ფსევდო-გადაწყვეტადი პროპოზიციული სქემებისათვისაც კი (მეტი ინფორმაციისათვის იხილეთ [ACP11]).

## 2.5 ალგორითმის რეალიზაცია

ალგორითმი განხორციელებულია GAPT [5] სისტემაში, რომლიც დაიწერა პროგრამირების ენა Scala-ზე [OSV10]. GAPT [EHR+16] სისტემაში რეალიზებულია მონაცემთა სტრუქტურები, ალგორითმები და მომხმარებლის ინტერფეისი, ფორმალური დამტკიცებების ანალიზისა და გარდაქმნისათვის. სისტემის სტრუქტურა არის ზოგადი და ახდენს ისეთი საბაზისო მონაცემთა სტრუქტურების რეალიზაციას, როგორიცაა ლამბადა კალკულუსი, სეკვენციათა და რეზოლუციის დამტკიცებები, გაფართოებული დამტკიცებები [HLRR13] და სხვა. სხვადასხვა ცნობილი თეორემათა დამამტკიცებლები უკვე ინტეგრირებულია ამ სისტემაში [DLL+12, HLR+14]. პარალელურად, ჩვენ შევიმუშავეთ მომხმარებლის გრაფიკული ინტერფეისი, რომელსაც ვუწოდებთ ProofTool, რომელიც გამოიყენება როგორც ვიზუალიზაციის ინსტრუმენტად (როგორიცა მაშტაბირება, გადაადგილება, ძებნა და ა.შ), აგრეთვე როგორც დამტკიცების მანიპულატორი (რომელიც საშუალებას გვაძლევს გამოვიძახოთ GAPT სისტემის დამტკიცების გარდაქმნის პროცედურები, როგორიცაა ქათ-ელიმინაცია, რეგულარიზაცია, სკოლემიზაცია და ა.შ. [6]). ProofTool რეალიზაციის დეტალები, თუ როგორ ახდენს ფორმულების, სეკვენციებისა და დამტკიცებების დარენდერებას ეკრანზე შესაძლებელია იხილოთ [DLL+13, LRR14]-ში.

GAPT სისტემაში რეალიზებულია რამდენიმე შეტანა/გამოტანის ფორმატები, მათ შორის ერთ-ერთია პირველი რიგის ფორმულათა სქემებისათვის. ეს ფორმატი არის ინტუიციური და მარტივი გამოსაყენებლად. მაგალითისათვის, ერთ-ერთი სეკვენცია ჩაწერილი ამ ფორმატით მოცემულია ფიგურა 2.2-ში. ჩვენ გვაქვს შემდეგი დაშვებები: მუდმივი ფუნქციონალური სიმბოლოები აღინიშნება `f, f1,...`, მუდმივი პრედიკატული სიმბოლოები `P, P1,...`, განსასაზღვრი ფუნქციონალური და პრედიკატული სიმბოლოები კი `g, g1,...` და `Q, Q1,...` შესაბამისად. ლოგიკური კავშირები

[5] General Architecture for Proof Theory, `http://www.logic.at/gapt`

[6] სისტემის ოფიციალური 2.0 ვერსიიდან მოყოლებული ეს ფუნქციონალი ამოღებულია გრაფიკული ინტერფეისიდან.

წარმოდგენილია ~, \/, /\, =>, Forall, Exists, ხოლო |- წარმოადგენს სეკვენციის ნიშანს. აღნიშნული ფორმატის დეტალური აღწერა დაინტერესებულ მკითხველს შეუძლია იხილოს [DLL+13]-ში.

GAPT სისტემაში რეალიზებულია LK კალკულუსის არაინვერსირებული ვერსია, მაგრამ გვაქვს ე.წ. მაკრო წესები, რომელიც აკეთებს ინვერსიული წესების სიმულაციას დამატებით შესუსტებისა და შეკვეცის წესების გამოყენებით. შესაბამისად, დამამტკიცებლიდან მიღებული სქემატური დამტკიცებები შეიცავს ზედმეტ სტრუქტურულ წესებს. GAPT-ში არსებობს ფუნქცია, რომელიც აშორებს ასეთ ზედმეტ წესებს დამტკიცებებიდან, მაგრამ ამჟამად ამ ფუნქიონალს არ ვიყენებთ, ვიანიდან ის მნიშვნელოვნად მოქმედებს დამამტკიცებლის წარმადობასა და სისწრაფეზე.

ალგორითმის რეალიზაცია შედგება რამდენიმე ობიექტისაგან. ერთ-ერთი მთავარი ობიექტია დამტკიცებათა ძებნის სტრატეგია, რომელიც განსაზღვრავს თუ რომელი წესები უნდა იქნას პირველად გამოყენებული და როგორ გამოიყენოს ესა თუ ის წესი. აღნიშნული ობიექტის შესაბამისი კოდის ფრაგმენტი მოცემულია ქვემოთ:

```
class SchemaProofStrategy extends ProofStrategy {
  val FormulaLocation = ProofStrategy.FormulaLocation // shortcut

  override def calcNextStep( seq: FSequent ):
      Option[ProofStrategy.Action] = {
    if ( SolveUtils.isAxiom( seq ) ||
        SolveUtils.findNonschematicAxiom( seq ).isDefined ) {
      throw new Exception( "Strategy called on axiom: " + seq )
    } else {
      // rule preference:
      // NOTE: getOrElse uses call by name, i.e.
      // functions below are only evaluated if really needed
      findUnaryRight( seq ).orElse(
```

```
      findUnaryLeft( seq ).orElse(
        findBinaryRight( seq ).orElse(
          findBinaryLeft( seq ).orElse(
            findStrongQuantifier( seq ).orElse(
              findWeakQuantifier( seq ).orElse( None )
                ) ) ) ) ) )
  }
}

// Tries to find a formula on the left or on the right
// such that its introduction rule is unary.
def findUnaryLeft( seq: FSequent ):
    Option[ProofStrategy.Action] =
  seq.antecedent.find( f => f match {
    case HOLNeg(_) | HOLAnd(_,_) | BigAnd(_,_,_,_) => true
    case _ => false
  } ).map( new ProofStrategy.Action(_,FormulaLocation.Antecedent,
                                    Some( this ) ) )

def findUnaryRight( seq: FSequent ):
    Option[ProofStrategy.Action] =
  seq.succedent.find( f => f match {
    case HOLNeg(_)|HOLImp(_,_)|HOLOr(_,_)|BigOr(_,_,_,_) => true
    case _ => false
  } ).map( new ProofStrategy.Action(_,FormulaLocation.Succedent,
                                    Some( this ) ) )

def findStrongQuantifier( seq: FSequent ):
    Option[ProofStrategy.Action] = {
  var qv: Option[HOLExpression] = None
```

```
  val antQ = seq.antecedent.find( f => f match {
    case HOLExVar( v, _ ) =>
      qv = Some( v )
      true
    case _ => false
  } ).map( new SchemaAction(_,FormulaLocation.Antecedent,
                            Some( this ), qv ) )
  antQ.orElse(
    seq.succedent.find( f => f match {
      case HOLAllVar( v, _ ) =>
        qv = Some( v )
        true
      case _ => false
    } ).map( new SchemaAction(_,FormulaLocation.Succedent,
                              Some( this ), qv ) ) )
}

def findWeakQuantifier( seq: FSequent ):
    Option[ProofStrategy.Action] = {
  val t = dbTRS.map.get( SchemaConst( "g",
          ->( Tindex, ->( Ti, Ti ) ) ) ).get._2._2
            .get( HOLPosition( 2 ) ).get
  val subst = SubstitutionHOL( t.get( HOLPosition( 2 ) )
              .get.asInstanceOf[HOLVar], HOLConst( "a", Ti ) )
  val term = subst( t )
  val antQ = seq.antecedent.find( f => f match {
    case HOLAllVar( _, _ ) =>
      true
    case _ => false
  } ).map( new SchemaAction(_,FormulaLocation.Antecedent,
```

```
                              Some( this ), Some( term ) ) )
  antQ.orElse(
    seq.succedent.find( f => f match {
      case HOLExVar( _, _ ) =>
        true
      case _ => false
    } ).map( new SchemaAction(_,FormulaLocation.Succedent,
                                Some( this ), Some( term ) ) ) )
}

// Tries to find a formula on the left or on the right
// such that its introduction rule is binary.
def findBinaryLeft( seq: FSequent ):
    Option[ProofStrategy.Action] =
  seq.antecedent.find( f => f match {
    case HOLImp(_,_) | HOLOr(_,_) | BigOr(_,_,_,_) => true
    case _ => false
  } ).map( new ProofStrategy.Action(_,FormulaLocation.Antecedent,
                                    Some( this ) ) )

def findBinaryRight( seq: FSequent ):
    Option[ProofStrategy.Action] =
  seq.succedent.find( f => f match {
    case HOLAnd(_,_) | BigAnd(_,_,_,_) => true
    case _ => false
  } ).map( new ProofStrategy.Action(_,FormulaLocation.Succedent,
                                    Some( this ) ) )

class SchemaAction( override val formula: HOLFormula,
           override val loc: ProofStrategy.FormulaLocation.Value,
```

```
                    val oldStrategy: Option[ProofStrategy],
                    val quantifiedTerm: Option[HOLExpression] )
      extends ProofStrategy.Action( formula, loc, oldStrategy ) {
    override def getQuantifiedTerm(): Option[HOLExpression] =
                                                    quantifiedTerm
  }
}
```

შემდეგი მნიშვნელოვანი ობიექტია solve ობიექტი, რომელიც იყენებს ზემოთ მოყვანილ სტრატეგიას და აგებს დამტკიცების სქემას. აღნიშნული ობიექტის შესაბამისი კოდის ფრაგმენტი მოცემულია ქვემოთ:

```
object solve {
  def solveSchema( seq: FSequent,
                   cleanStructuralRules: Boolean = false,
                   throwOnError: Boolean = true ):
      Option[LKProof] = {
    if ( SolveUtils.noCommonAtoms( seq ) ) None
    startProving( seq, new SchemaProofStrategy,
                  cleanStructuralRules, throwOnError )
  }

  private def startProving( seq: FSequent, strategy: ProofStrategy,
            cleanStructuralRules: Boolean, throwOnError: Boolean ):
      Option[LKProof] = {
    val seq_norm = FSequent( seq.antecedent.toSet.toList,
                             seq.succedent.toSet.toList )
    prove( seq_norm, strategy ) match {
      case Some( p ) => {
```

```
        val pWithWeakening = WeakeningMacroRule( p, seq )
        Some( if ( cleanStructuralRules )
                CleanStructuralRules( pWithWeakening )
              else pWithWeakening )
      }
      case None => {
        if ( throwOnError ) {
          throw new Exception( "Sequent is not provable." )
        } else {
          None
        }
      }
    }
  }

  private def prove( seq: FSequent, strategy: ProofStrategy ):
      Option[LKProof] = {
    val ant_set = seq.antecedent.toSet
    val suc_set = seq.succedent.toSet
    if ( SolveUtils.isAxiom( seq ) ) {
      val ( f, rest ) = SolveUtils.getAxiomfromSeq( seq )
      val p = WeakeningMacroRule( Axiom( f::Nil, f::Nil ), seq )
      Some( p )
    }
    else if ( SolveUtils.findNonschematicAxiom( seq ).isDefined ) {
      val Some( (f,g) ) = SolveUtils.findNonschematicAxiom( seq )
      Some( AtomicExpansion( seq, f, g ) )
    }
    else if ( strategy.isInstanceOf[SchemaProofStrategy] &&
              SolveUtils.findSchematicAxiom( seq ).isDefined ) {
```

```
      SolveUtils.findSchematicAxiom( seq )
    }
    else if ( strategy.isInstanceOf[SchemaProofStrategy] &&
              SolveUtils.findProofLink( seq ).isDefined ) {
      SolveUtils.findProofLink( seq )
    }
    else {
      // main step: ask strategy what to do
      strategy.calcNextStep( seq ) match {
        case Some( action ) => {
          // dumbly apply whatever rule matches to this formula
          action.loc match {
            case ProofStrategy.FormulaLocation.Antecedent =>
              assert( seq.antecedent.contains( action.formula ) )
              applyActionAntecedent( action, seq )
            case ProofStrategy.FormulaLocation.Succedent =>
              assert( seq.succedent.contains( action.formula ) )
              applyActionSuccedent( action, seq )
          }
        }
        case None => None
      }
    }
  }

  // auxiliary function to bypass rule application if possible
  private def trySkipRuleApplication( toLeft: List[HOLFormula],
                                      toRight: List[HOLFormula] ):
      Option[LKProof] = {
    SolveUtils.canSkipRuleApplication( toLeft, toRight, seq ) ?
```

```
    prove( rest, nextProofStrategies( 0 ) )
      .map( WeakeningLeftRule( _, action.formula ) )
  : None
}

private def applyActionAntecedent( action: ProofStrategy.Action,
                                   seq: FSequent ):
    Option[LKProof] = {
  // sequent without principal formula to build upper sequent
  val rest = FSequent( seq.antecedent.diff(action.formula::Nil),
                       seq.succedent )
  // proof strategies for children
  val nextProofStrategies = action.getNextStrategies()
  action.formula match {
    // case distinction on logical connectives
    // to build corresponding upper sequent and its proof
    case HOLAllVar( v, f ) => {
      val quantifiedTerm = action.getQuantifiedTerm().get
      val auxFormula = SubstitutionHOL( v, quantifiedTerm )( f )
      val p_ant = action.formula +: auxFormula +: rest.antecedent
      val p_suc = rest.succedent
      val premise = FSequent( p_ant, p_suc )
      prove( premise, nextProofStrategies( 0 ) ).map( proof => {
        // weak quantifier, keep formula
        val p1 = ForallLeftRule( proof, auxFormula,
                                 action.formula, quantifiedTerm )
        ContractionLeftRule( p1, action.formula )
      } )
    }
    case HOLExVar( v, f ) => {
```

```
        val eigenVar = action.getQuantifiedTerm().get
        val auxFormula = SubstitutionHOL( v, eigenVar )( f )
        val p_ant = auxFormula +: rest.antecedent
        val p_suc = rest.succedent
        val premise = FSequent( p_ant, p_suc )
        prove( premise, nextProofStrategies( 0 ) ).map( proof =>
          ExistsLeftRule(proof,auxFormula,action.formula,eigenVar))
      }
      case HOLNeg( f1 ) =>
        trySkipRuleApplication( Nil, f1 :: Nil ).orElse( {
          // Computing premise antecedent and succedent
          val p_ant = rest.antecedent
          val p_suc = f1 +: rest.succedent
          val premise = FSequent( p_ant, p_suc )
          prove( premise, nextProofStrategies( 0 ) ) match {
            case Some( p ) => Some( NegLeftRule( p, f1 ) )
            case None      => None
          }
        } )
      case HOLAnd( f1, f2 ) =>
        trySkipRuleApplication( f1 :: f2 :: Nil, Nil ).orElse( {
          // If one formula is there, do not contract
          if ( SolveUtils.checkDuplicate( f1::Nil, Nil, seq ) ) {
            val up_ant = f2 +: rest.antecedent
            val up_suc = rest.succedent
            val upremise = FSequent( up_ant, up_suc )
            prove( upremise, nextProofStrategies(0) ).map( proof =>
              AndLeft2Rule( proof, f1, f2 ) )
          }
          else if ( SolveUtils.checkDuplicate(f2::Nil,Nil,seq ) ) {
```

```
          val up_ant = f1 +: rest.antecedent
          val up_suc = rest.succedent
          val upremise = FSequent( up_ant, up_suc )
          prove( upremise, nextProofStrategies(0) ).map( proof =>
            AndLeft1Rule( proof, f1, f2 ) )
        } else {
          val up_ant = f1 +: f2 +: rest.antecedent
          val up_suc = rest.succedent
          val upremise = FSequent( up_ant, up_suc )
          prove( upremise, nextProofStrategies( 0 ) ) match {
            case Some( proof ) =>
              val proof_and2 = AndLeft2Rule( proof, f1, f2 )
              val proof_and1 = AndLeft1Rule( proof_and2, f1, f2 )
              val proof_contr =
                ContractionLeftRule( proof_and1, action.formula )
              Some( proof_contr )
            case None => None
          }
        }
      } )
    case HOLImp( f1, f2 ) =>
      trySkipRuleApplication( f2 :: Nil, Nil ).orElse(
        trySkipRuleApplication( Nil, f1 :: Nil ).orElse( {
          val p_ant1 = rest.antecedent
          val p_suc1 = f1 +: rest.succedent
          val p_ant2 = f2 +: rest.antecedent
          val p_suc2 = rest.succedent
          val premise1 = FSequent( p_ant1, p_suc1 )
          val premise2 = FSequent( p_ant2, p_suc2 )
          prove( premise1, nextProofStrategies( 0 ) ) match {
```

```
            case Some( p1 ) => prove( premise2,
                          nextProofStrategies( 1 ) ) match {
              case Some( p2 ) =>
                val p = ImpLeftRule( p1, p2, f1, f2 )
                val p_contr =
                  ContractionMacroRule( p, seq, strict = false )
                Some( p_contr )
              case None => None
            }
            case None => None
          }
        } ) )
    case HOLOr( f1, f2 ) =>
      trySkipRuleApplication( f1 :: Nil, Nil ).orElse(
        trySkipRuleApplication( f2 :: Nil, Nil ).orElse( {
          val p_ant1 = f1 +: rest.antecedent
          val p_suc1 = rest.succedent
          val p_ant2 = f2 +: rest.antecedent
          val p_suc2 = rest.succedent
          val premise1 = FSequent( p_ant1, p_suc1 )
          val premise2 = FSequent( p_ant2, p_suc2 )
          prove( premise2, nextProofStrategies( 0 ) ) match {
            case Some( p2 ) => prove( premise1,
                          nextProofStrategies( 1 ) ) match {
              case Some( p1 ) =>
                val p = OrLeftRule( p1, p2, f1, f2 )
                val p_contr =
                  ContractionMacroRule( p, seq, strict = false )
                Some( p_contr )
              case None => None
```

```
            }
            case None => None
          }
        } ) )
    case _ => throw new IllegalArgumentException(
                "Invalid formula in prove: " + action.formula )
  }
}

private def applyActionSuccedent( action: ProofStrategy.Action,
                                  seq: FSequent ):
    Option[LKProof] = {
  // sequent without principal formula to build upper sequent
  val rest = FSequent( seq.antecedent,
                       seq.succedent.diff( action.formula::Nil ))
  // proof strategies for children
  val nextProofStrategies = action.getNextStrategies()
  action.formula match {
    // case distinction on logical connectives
    // to build corresponding upper sequent and its proof
    case HOLAllVar( v, f ) => {
      val eigVar = action.getQuantifiedTerm().get
      val auxFormula = SubstitutionHOL( v, eigenVar )( f )
      val p_ant = rest.antecedent
      val p_suc = auxFormula +: rest.succedent
      val premise = FSequent( p_ant, p_suc )
      prove( premise, nextProofStrategies( 0 ) ).map( proof =>
        ForallRightRule(proof,auxFormula,action.formula,eigVar))
    }
    case HOLExVar( v, f ) => {
```

```
      val quantifiedTerm = action.getQuantifiedTerm().get
      val auxFormula = SubstitutionHOL( v, quantifiedTerm )( f )
      val p_ant = rest.antecedent
      val p_suc = action.formula +: auxFormula +: rest.succedent
      val premise = FSequent( p_ant, p_suc )
      prove( premise, nextProofStrategies( 0 ) ).map( proof => {
        // weak quantifier, keep formula
        val p1 = ExistsRightRule( proof, auxFormula,
                                  action.formula, quantifiedTerm)
        ContractionRightRule( p1, action.formula )
      } )
    }
    case HOLNeg( f1 ) =>
      trySkipRuleApplication( f1 :: Nil, Nil ).orElse( {
        val p_ant = f1 +: rest.antecedent
        val p_suc = rest.succedent
        val premise = FSequent( p_ant, p_suc )
        prove( premise, nextProofStrategies( 0 ) ).map( p =>
          NegRightRule( p, f1 ) )
      } )
    case HOLImp( f1, f2 ) =>
      trySkipRuleApplication( f1 :: Nil, f2 :: Nil ).orElse( {
        val p_ant = f1 +: rest.antecedent
        val p_suc = f2 +: rest.succedent
        val premise = FSequent( p_ant, p_suc )
        prove( premise, nextProofStrategies( 0 ) ) match {
          case Some( p ) =>
            val p1 = ImpRightRule( p, f1, f2 )
            Some( p1 )
          case None => None
```

```
        }
      } )
    case HOLOr( f1, f2 ) =>
      trySkipRuleApplication( Nil, f1::f2::Nil ).orElse( {
        if ( SolveUtils.checkDuplicate( Nil, f2::Nil, seq ) ) {
          val up_ant = rest.antecedent
          val up_suc = f1 +: rest.succedent
          val upremise = FSequent( up_ant, up_suc )
          prove( upremise, nextProofStrategies(0) ).map( proof =>
            OrRight1Rule( proof, f1, f2 ) )
        }
        else if ( SolveUtils.checkDuplicate(Nil,f1::Nil,seq) ) {
          val up_ant = rest.antecedent
          val up_suc = f2 +: rest.succedent
          val upremise = FSequent( up_ant, up_suc )
          prove( upremise, nextProofStrategies(0) ).map( proof =>
            OrRight2Rule( proof, f1, f2 ) )
        } else {
          val up_ant = rest.antecedent
          val up_suc = f1 +: f2 +: rest.succedent
          val upremise = FSequent( up_ant, up_suc )
          prove( upremise, nextProofStrategies( 0 ) ) match {
            case Some( proof ) =>
              val proof_or2 = OrRight2Rule( proof, f1, f2 )
              val proof_or1 = OrRight1Rule( proof_or2, f1, f2 )
              val proof_contr =
                ContractionRightRule( proof_or1, action.formula )
              Some( proof_contr )
            case None => None
          }
```

```
          }
        } )
      case HOLAnd( f1, f2 ) =>
        trySkipRuleApplication( Nil, f1 :: Nil ).orElse(
          trySkipRuleApplication( Nil, f2 :: Nil ).orElse( {
            val p_ant1 = rest.antecedent
            val p_suc1 = f1 +: rest.succedent
            val p_ant2 = rest.antecedent
            val p_suc2 = f2 +: rest.succedent
            val premise1 = FSequent( p_ant1, p_suc1 )
            val premise2 = FSequent( p_ant2, p_suc2 )
            prove( premise2, nextProofStrategies( 0 ) ) match {
              case Some( p2 ) => prove( premise1,
                                 nextProofStrategies( 1 ) ) match {
                case Some( p1 ) =>
                  val p = AndRightRule( p1, p2, f1, f2 )
                  val p_contr =
                    ContractionMacroRule( p, seq, strict=false )
                  Some( p_contr )
                case None => None
              }
              case None => None
            }
          } ) )
      case _ => throw new IllegalArgumentException(
                  "Invalid formula in prove: " + action.formula )
    }
  }
}
```

ამჟამად დამამტკიცებელი პოულობს მხოლოდ “მატივ” დამტკიცების სქემებს. “მარტივ”-ში იგულისხმება დამტკიცებათა სქემა, რომელიც შედგება მხოლოდ ერთი მტკიცებათა წყვილისაგან. ეს ნიშნავს, რომ რეკურსული შემთხვევის LKS-დამტკიცებაში არსებული მტკიცებათა აქსიომები მიუთითებს ისევ ამ დამტკიცებათა წყვილის სიმბოლოზე. შემდეგი მაგალითი აჩვენებს ამ პრობლემას:

**მაგალითი 2.5.1.** განვიხილოთ მაგალითი 2.3.3-ში მოცემული სეკვენციის მარტივი მოდიფიკაცია:

$$P(a), \forall x(\neg P(x) \vee P(f(x))) \vdash P(g(k,x)) \wedge P(a).$$

ჩვენი დამტკიცების პროცედურა, საბაზისო შემთხვევისათვის აგებს ტრივიალურ დამტკიცებას, მაგრამ ვერ ახერხებს რეკურსიული ნაწილის აგებას. ალგორითმის თანახმად, პირველი დაიშლება სეკვენციის მარჯვენა მხარეს არსებული ფორმულა და მივიღებთ შემდეგს (სტრუქტურული წესები გამოტოვებულია):

$$\frac{P(a), \forall x(\neg P(x) \vee P(f(x))) \vdash P(g(k,x)) \quad P(a) \vdash P(a)}{P(a), \forall x(\neg P(x) \vee P(f(x))) \vdash P(g(k,x)) \wedge P(a)} \wedge_{\mathsf{r}}$$

ამ ეტაპზე დამტკიცება ცდილობს სეკვენციის მარცხენა მხარეს არსებული ფორმულის დაშლას და შედეგად ვერ ახერხებს თეორემის დამტკიცებას. დამამტკიცებლის სწორი მოქმედება კი იქნებოდა, რომ შემოეღო ახალი დამტკიცების აქსიომა და მოეძებნა დამტკიცებათა წყვილი სეკვენციისათვის $P(a), \forall x(\neg P(x) \vee P(f(x))) \vdash P(g(k,x))$. სამომავლოდ იგეგმება ამ პრობლემაზე მუშაობა.

ბოლოს, ჩვენ გვინდა აღვნიშნოთ, რომ დამამტკიცებელი ინტეგრირებულია ProofTool-ში. Prover>Start მენიუ იძახებს ახალ ფანჯარას, სადაც მომხმარებელს შეუძლია შეიყვანოს დასამტკიცებელი სეკვენციის სქემა (იხილეთ ფიგურა 2.2). აღნიშნული მენიუს შესაქმნელი კოდის ფრაგმენტი მოცემულია ქვემოთ:

```
contents += new Menu( "Prover" ) {
  contents += new MenuItem( Action( "Start" ) {
    SchemaProver()
  } ) { border = customBorder }
}
```

Parse ღილაკზე დაჭერის შემდეგ შეტანილი ტექსტი გარდაიქმნება შესაბამის სეკვენციათა სქემის ობიექტად და გამოიძახება დამტკიცებათა სქემის აგების ალგორითმი. თუ საბაზისო დამტკიცება იპოვნა, იგი გამოაქვს ეკრანზე (იხილეთ ფიგურა 2.3), და აგრძელებს რეკურსული დამტკიცების ძებნას. თუ ასეთი დამტკიცება მოიძებნა, მაშინ ისიც გამოდის ეკრანზე (იხილეთ ფიგურა 2.4). წინააღმდეგ შემთხვევაში გამოვა შეტყობინება შეცდომის შესახებ. აღწერილი პროცესის შესაბამისი კოდის ნაწილი (SchemaProver ობიექტი) მოცემულია ქვემოთ:

```
object SchemaProver {
  def apply(): Unit = try {
    Main.body.cursor =
        new java.awt.Cursor( java.awt.Cursor.WAIT_CURSOR )
    val t = TextAreaDialog( "Please enter a sequent schema:" )
    if ( t != None && t.get != "" ) {
      val s = parse( t.get )
      prove( s )
    }
  } finally {
    Main.body.cursor = java.awt.Cursor.getDefaultCursor
  }
```

```
def parse( t: String ): FSequent = try {
  val sequent = sFOParser.parseSequent( t )
  val defs = dbTRS.map.map( p => p._2._1 :: p._2._2 :: Nil )
    .flatten.toMap[HOLExpression, HOLExpression]
  Main.db.addDefinitions( defs )
  ProofToolPublisher.publish( ProofDbChanged )
  sequent
} catch {
  case e: Throwable =>
    throw new Exception( "Cannot parse the sequent!\n\n"
      + Main.getExceptionString( e ) )
}

def prove( s: FSequent ): Unit = {
  val k = IntVar( "k" )
  val sp = new SchemaProof( "\\psi", k :: Nil, s, null, null )
  SchemaProofDB.clear
  SchemaProofDB.put( sp )
  val subst_0 = SchemaSubstitution( ( k, IntZero() ) :: Nil )
  val s_0 = FSequent( s.antecedent.map( fo => subst_0( fo ) ),
                      s.succedent.map( fo => subst_0( fo ) ) )
  val subst_k = SchemaSubstitution( ( k, Succ( k ) ) :: Nil )
  val s_k = FSequent( s.antecedent.map( fo => subst_k( fo ) ),
                      s.succedent.map( fo => subst_k( fo ) ) )
  Main.db.addSeqList( List( s_0, s_k ) )

  try {
    val p_base = try {
      solve.solveSchema( sp.b_res )
    } catch {
```

```
        case e: Throwable =>
          Main.errorMessage( "Cannot prove the base case!\n\n"
            + Main.getExceptionString( e ) )
          throw new Exception( "Terminating with None" )
      }

      Main.db.addProofs( ( "\\psi_base", p_base.get ) :: Nil )
      ProofToolPublisher.publish( ProofDbChanged )
      Main.updateLauncher( "\\psi_base", p_base.get, 16 )
      Main.questionMessage( "Proof of the base case found.\r\n
        Start proving the step case" ) match {
          case Dialog.Result.Yes =>
            val p_step = try {
              solve.solveSchema( sp.r_res )
            } catch {
              case e: Throwable =>
                Main.errorMessage("Cannot prove the step case!\n\n"
                  + Main.getExceptionString( e ) )
                throw new Exception( "Terminating with None" )
            }
            Main.db.addProofs( ( "\\psi_step", p_step.get )::Nil )
            Main.updateLauncher( "\\psi_step", p_step.get, 16 )
            ProofToolPublisher.publish( ProofDbChanged )
          case _ =>
      }
    } catch {
      case e: Throwable => println( e.getMessage )
    }
  }
}
```

მტკიცებათა სქემების აგების ზემოთ მოყვანილი ალგორითმი და მისი რეალიზაციის შესახებ ინფორმაცია პირველად გამოქვეყნებულია სტატიაში [CKR16b]. ამ ნაშრომში მოცემულია ალგორითმის რეალიზაციის გაუმჯობესებული ვარიანტი, სადაც ჩასწორებულია რამდენიმე შეცდომა. როგორც ზემოთ იყო ნახსენები, ამ მომენტისათვის GAPT სისტემა სხვა მიმართულებით ვითარდება, შესაბამისად ჩვენი ალგორითმი ამ ფორმით არ არის ინტეგრირებული მის უახლეს ვერსიებში. ჩვენი სისტემის ჩამოტვირთვა შესაძლებელია მისამართიდან `http://www.viam.science.tsu.ge/projects/fr_51_4-102_13/lks_prover.zip`.

# თავი 3

# თეორემათა მტკიცება ურანგო ლოგიკაში

ეს თავი განკუთვნილია ისეთი ძლიერი გამომსახველობის მქონე ფორმალიზმებსათვის, რომლებიც დაფუძნებულია ურანგო ალფაბეტზე, ანუ ფუნქციონალურ ან/და პრედიკატულ სიმბოლოებს დაფიქსირებული ადგილიანობა არ აქვთ, ძირითადად მათი გამოყენების საკმაოდ ფართო სფეროს გამო. ასეთ ენების საშუალებით ბუნებრივად შეიძლება XML დოკუმენტების და მათზე ოპერაციების მოდელირება, კრიპტოგრაფიული პროტოკოლების და პარალელური პროცესების ანალიზი და სხვა. ასეთ ენებს ურანგო ენებს უწოდებენ.

ურანგო ტერმები წარმოადგენენ პირველი რიგის ტერმებს, სადაც ფუნქციონალურ სიმბოლოებს არ აქვთ ფიქსირებული ადგილიანობა: მსგავს სიმბოლოებს შეიძლება ჰქონდეთ არგუმენტების სხვადასხვა რაოდენობა სხვადასხვა ადგილას და ზოგიერთ ცვლადები (მიმდევრობითი ცვლადები) შესაძლებელია წარმოვადგინოთ სასრული (შესაძლებელია ცარიელი) ურანგო ტერმების მიმდევრობად.

ბოლო წლებში, მიმდევრობითი ცვლადების და ურანგო სიმბოლოების გამოყენება ილუსტრირებულია პრაქტიკულ XML-ის აპლიკაციებში [CF04, KM05, CF06, CF07, CFK07, CFK09], სქემების გარდაქმნის ოპერაციებში [RF97, CD97], ცოდნის წარმოდგენაში [Gen98, HM05, HM01b, Men11], ავტომატური მსჯელობისას [Pau90, Gin91, Kut03, HV06, BCJ+06], გადაწერაში [Ham97, WB01, JR08], ფუნქციონალურ, ლოგიკურ და წესებზე დაფუძნებულ პროგრამირებაში [MT03, Bol99, MK03, MK06], და სხვა. არსებობს სისტემები, რომლებიც იყენებენ მიმდევრობით

ცვლადებს პროგრამირებაში. ასეთი ყველაზე ცნობილი სისტემა არის Mathematica [Wol03], რომელსაც აქვს საკმაოდ მძლავრი წესებზე დაფუძნებული პროგრამირების ენა და იყენებს ისეტ ტექნიკებს, როგორიცაა (პირველი რიგის, ტოლობიანი) ურანგო შეთანადება მიმდევრობითი ცვლადებით [Buc96]. ურანგო ფუნქციონალური სიმბოლოები და მიმდევრობითი ცვლადები ამ ენას ანიჭებენ ძალიან დიდ გამომსახველობით უნარს, რაც იძლევა საშუალებას, რომ დაიწეროს მოკლე, კონკრეტული და წაკითხვადი კოდი.

## 3.1 ურანგო სეკვენციათა კალკულუსი LKU

განვიხილოთ ალფავიტი $\mathcal{A}$, რომელიც შედგება შემდეგი სიმბოლოებისგან:

- $\mathcal{V}_\mathsf{I}$ -ინდივიდუალურ ცვლადთა სიმრავლე, აღინიშნება $x, y, z, ...$,
- $\mathcal{V}_\mathsf{S}$ -მიმდევრობით ცვლადთა სიმრავლე, აღინიშნება $\overline{x}, \overline{y}, \overline{z}, ...$,
- $\mathcal{F}_r$ -ფიქსირებულ ადგილიანი (რანგიანი) ფუნქციონალურ სიმბოლოთა სიმრავლე, აღინიშნება $f_r$,
- $\mathcal{F}_u$ - მოძრავ ადგილიანი (ურანგო) ფუნქციონალურ სიმბოლოთა სიმრავლე, აღინიშნება $f_u$,
- $\mathcal{P}_r$ -რანგიანი პრედიკატული სიმბოლოთა სიმრავლე, აღინიშნება $p_r$,
- $\mathcal{P}_u$ - ურანგო პრედიკატული სიმბოლოთა სიმრავლე, აღინიშნება $p_u$,
- პროპოზიციული მუდმივები true და false,
- ლოგიკური კავშირები და კვანტორები $\neg, \vee, \wedge, \rightarrow, \exists, \forall$,
- დამხმარე სიმბოლოები:ფრჩხილები და მძიმე.

თითოეულ რანგიან სიმბოლოს აქვს უნიკალური ადგილი (რანგი), რომელიც ასოცირებულია მასთან. ურანგო სიმბოლოებს არ აქვთ

ფიქსირებული ადგილიანობა. თითოეული ცვლადთა სიმრავლე, ფუნქციონალური სიმბოლოები და პრედიკატულლი სიმბოლოები არიან სასრული. $\mathcal{F}$-ით აღვნიშნავთ ფუნქციონალურ სიმბოლოების სიმრავლეს და ასევე $\mathcal{F}_u$ და $\mathcal{F}_r$ სიმრავლეთა გაერთიანებას. შესაბამისად $\mathcal{P}$ აღნიშნავს პრედიკატული სიმბოლოების სიმრავლეს და ასევე $\mathcal{P}_u$ და $\mathcal{P}_r$ სიმრავლეთა გაერთიანებას. ჩვენ ვიყენებთ $f$-ს რომ აღვნიშნოთ $\mathcal{F}$-ის ელემენტები, და $p$-ს რომ აღვნიშნოთ $\mathcal{P}$-ის ელემენტები, ხოლო $\mathcal{V}$ აღნიშნავს სიმრავლეს $\mathcal{V}_\mathsf{I} \cup \mathcal{V}_\mathsf{S}$.

*ტერმები* და *მიმდევრობები* $\mathcal{A}$-ზე განისაზღვრება ინდუქციური გზით შემდეგნაირად:

$$
\begin{array}{ll}
t ::= x \mid f_r(t_1, \ldots, t_n) \mid f_u(\overline{s}) & \text{თერმები} \\
\overline{s} ::= t \mid \overline{x} \mid (\overline{s}_1, \ldots, \overline{s}_n) \quad (n \geq 0) & \text{მიმდევრობები}
\end{array}
$$

*ატომი* არის ფორმულა შემდეგი ფორმის $p_r(t_1, \ldots, t_n)$ და $p_u(\overline{s})$, სადაც $p_r \in \mathcal{P}_r$ არის $n$-ური პრედიკატული სიმბოლო და $p_u \in \mathcal{P}_u$ არის ფიქსირებულადგილიანი პრდიკატული სიმბოლო. ჩვენ ვიყენებთ შემდეგნაირ აღნიშვნას ატომებისთვის $A$. *ფორმულები* აგებულია სტანდარტულად ატომალური ფორმულებისგან და ლოგიკური ოპერატორებისგან $\neg, \vee, \wedge, \rightarrow$ ,$\exists$ და $\forall$. კვანტორები დაშვებულია ორივეზე, როგორც მიმდევრობით ცვლადებზე ასევე ინდივიდუალურ ცვლადებზე.

[KB04]-ის მიხედვით, ჩვენ განვსაზღვრეთ სეკვენციათა კალკულუსი LKU, რომელიც მიიღება LK კალკულუსზე ისეთი კვანტორების წესის დამატებით, რომლებიც მოქმედებენ მიმდევრობით ცვლადებზე (იხილეთ ფიგურა 3.1).

ურანგო ტერმების უნიფიკაციის პროცედურა მოცემულია [Kut07]. ურანგო ტერმები $f(\overline{x}, x, \overline{y}, \overline{z})$ და $f(f(\overline{x}), x, a)$, სადაც $\overline{x}, \overline{y}, \overline{z}$ მიმდევრობითი ცვლადებია და $x$ -არის ინდივიდუალური ცვლადი, უნიფიკატორების ჩასმა ხდება სამი სხვადასხვა გზით $\{\overline{x} \mapsto (), x \mapsto f(), \overline{y} \mapsto (), \overline{z} \mapsto (f(), a)\}$, $\{\overline{x} \mapsto (), x \mapsto f(), \overline{y} \mapsto f(), \overline{z} \mapsto a\}$ და $\{\overline{x} \mapsto (), x \mapsto f(), \overline{y} \mapsto (f(), a), \overline{z} \mapsto ()\}$. შევნიშნოთ, რომ ურანგო ტერმებს შეიძლება ჰქონდეთ უსასრულოდ ბევრი უნიფიკატორი. მაგალითად: $f(\overline{x}, a)$ და $f(\overline{x}, a)$ აქვს შემდეგი უნიფიკატორები $\{\overline{x} \mapsto ()\}$, $\{\overline{x} \mapsto a\}$, $\{\overline{x} \mapsto (a, a)\}$, და ა.შ. ურანგო უნიფიკაცია, როდესაც ერთი

LKU $= (\mathsf{LKS} \setminus \{\mathsf{pax}, \mathcal{E}\}) \cup \{\forall^{\mathsf{u}}_{\mathsf{l}}, \forall^{\mathsf{u}}_{\mathsf{r}}, \exists^{\mathsf{u}}_{\mathsf{l}}, \exists^{\mathsf{u}}_{\mathsf{r}}, \approx\}$, სადაც:

$$\frac{A(\ulcorner t_1, \ldots, t_n \urcorner), \Gamma \vdash \Delta}{\forall \bar{x} A(\bar{x}), \Gamma \vdash \Delta} \forall^{\mathsf{u}}_{\mathsf{l}} \qquad \frac{\Gamma \vdash \Delta, A(\bar{v})}{\Gamma \vdash \Delta, \forall \bar{x} A(\bar{x})} \forall^{\mathsf{u}}_{\mathsf{r}}{}^{1}$$

$$\frac{A(\bar{v}), \Gamma \vdash \Delta}{\exists \bar{x} A(\bar{x}), \Gamma \vdash \Delta} \exists^{\mathsf{u}}_{\mathsf{l}}{}^{1} \qquad \frac{\Gamma \vdash \Delta, A(\ulcorner t_1, \ldots, t_n \urcorner)}{\Gamma \vdash \Delta, \exists \bar{x} A(\bar{x})} \exists^{\mathsf{u}}_{\mathsf{r}}$$

$$\frac{\Gamma, (s \approx t \wedge a[s]) \Rightarrow a[t] \vdash \Delta}{\Gamma \vdash \Delta} \approx$$

[1] $\bar{v}$ არის მიმდევრობითი ცვლადი, რომელიც არ გვხვდება $\Gamma, \Delta, A(\bar{x})$.

ფიგურა 3.1: ურანგო სეკვენციათა კალკულუსი LKU.

ტერმი არის შემოსაზღვრული (ცვლადების გარეშე) არის სასრული.

### 3.1.1 შესაბამისობა ურანგო ლოგიკასა და ფორმულათა სქემებს შორის

ურანგო ფორმულა რომ ჩავწეროთ როგორც ფორმულათა სქემა, საკმარისია მიმდევრობითი ტერმები ჩავწეროთ ტერმთა სქემებად. დანარჩენი გარდაქმნები არის პირდაპირი.

ვთქვათ $\ulcorner t_1, \ldots, t_n \urcorner$ არის მიმდევრობითი ტერმი. ვთქვათ $g$ არის ბინარული ფუნქციონალური სიმბოლო. მაშინ $\ulcorner t_1, \ldots, t_n \urcorner \overset{def}{\equiv} g(\ldots g(t_1, t_2), \ldots), t_n)$ და ჩვენ შეგვიძლია ავაგოთ შესაბამისი ტერმთა სქემა:

$$\hat{g}(1) \to t_1 \qquad \text{და} \qquad \hat{g}(n+1) \to g(\hat{g}(n), t_{n+1})$$

ასევე შესაძლებელია LKU-გამოყვანა გარდავქმნათ LKS-გამოყვანად. ურანგო კვანტორების წესი დაემთხვევა LKS კვანტორების წესს მიმდევრობითი თერმების ტერმთა სქემებად გარდაქმნის შემდეგ. დაგვრჩა მხოლოდ $\approx$ წესის გარდავქმათ LKS-გამოყვანად.

[DLRW15, Ruk13]-ში ნაჩვენებია, რომ LKS $\equiv$ LKS $\cup$ {cut}, სადაც

განკვეთის წესი არის :

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Pi \vdash \Lambda}{\Gamma, \Pi \vdash \Delta, \Lambda}$$

$\approx$ წესი შეისაძლებელია მივიღოთ LKS $\cup$ {cut}-ში, შესაბამისად, არეთვე LKS-შიც. $\approx$ წესის გამოყვანა LKS $\cup$ {cut} მოცემულია შემდეგნაირად:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a[s] \vdash a[s]}{s \approx t, a[s] \vdash a[s]}\mathsf{w_l}}{s \approx t, a[s] \vdash a[t]}\mathcal{E}}{s \approx t \wedge a[s] \vdash a[t]}\wedge_\mathsf{l}}{\vdash (s \approx t \wedge a[s]) \Rightarrow a[t]}\Rightarrow_\mathsf{r} \quad \Gamma, (s \approx t \wedge a[s]) \Rightarrow a[t] \vdash \Delta}{\Gamma \vdash \Delta}\mathsf{cut}$$

ამრიგად, ყოველი ურანგო ფორმულა შესაძლებელია წარმოვადგინოთ ფორმულათა სქემის სახით. მიუხედავად იმისა, რომ [KB04]-ში ავტორებმა მოიყვანეს ინდუქციის წესი ურანგო ფორმულებისათვის, ფორმულათა სქემის წარმოდგენა ურანგო ფორმულის სახით შეუძლებლად გვეჩვენება. აღნიშნული შედეგები გამოქვეყნებულია [KR15]-ში.

ამ ორ ფორმალიზმს შორის შესაბამისობის ანალიზი დაგვეხმარა პარაგრაფ 2.4-ში მოყვანილი ალგორითმის გაგვრცობაში აგრეთვე LKU კალკულუსისათვის. აღნიშნული გაფართოების რეალიზაციის დეტალები აღწერილია შემდეგ პარაგრაფში, რომელიც პირველად გამოქვეყნებული იყო [CKR16a]-ში.

### 3.1.2 დამტკიცებათა აგების ალგორითმის რეალიზაცია

ჩვენს მიერ რეალიზებული სისტემა[1] წარმოადგენს მრავალდონიან აპლიკაციას, რომელიც შედგება ორი ნაწილისაგან. დამტკიცებათა აგების ალგორითმი (რომელსაც ქვემოთ აღვწერთ) დაწერილია Transact SQL ენაზე. ჩვენ SQL ავირჩიეთ რამდენიმე მიზეზის გამო: ჯერ აქამდე არავის უცდია ასეთი პროცედურის დაწერა SQL ენაზე, თუმცა, არ არსებობს ამის რაიმე მძლავრი მიზეზი. Transact SQL ენა არის ტიურინგ-სრული, რაც ნიშნავს, რომ

[1] სისტემის ჩამოტვირთვა შესაძლებელია მისამართიდან `http://www.viam.science.tsu.ge/projects/fr_51_4-102_13/lku_prover.zip`

მას გააჩნია ისეთივე გამომსახველობითი უნარი, რაც სხვა პროგრამირების ენებს. თვითონ SQL ბირთვი საკმაოდ სწრაფია, როდესაც სჭირდება დიდი რაოდენობა ობიექტების დამუშავება, რაც ხშირად ხდება თეორემათა დამტკიცებისას.

მეორე მიზეზი უკავშირდება პროცედურის პრაქტიკულ გამოყენებას – სამომავლოდ ვგეგმავთ რელაციურ მონაცემთა ბაზებში შენახული დიდი მონაცემების დამუშავებას ჩვენი სისტემის საშუალებით. ჩვენ ვფიქრობთ, რომ ასეთი პროცედურის ქონა მონაცემთა ბაზის შიგნით გააუმჯობესებს წარმადობას. აგრეთვე, SQL არის პლატფორმისაგან დამოუკიდებელი, რაც გულისხმობს იმას, რომ მასზე შესაძლებელია ნებისმიერი ტიპის ინტერფეისის (დესკტოპი, ვები, და ა.შ.) მიბმა.

სისტემის მეორე ნაწილს წარმოადგენს მომხმარებლის გრაფიკული ინტერფეისი (ე.წ. GUI), რომელიც დაწერილია Visual Prolog ენაზე. გრაფიკული ინტერფეისი დაკავშირებულია მონაცემთა ბაზასთან, აწვდის მას დასამტკიცებელ სეკვენციას, მონაცემთა ბაზიდან იღებს დამტკიცების ინფორმაციას (როგორიცაა გამოყენებული წესების მიმდევრობა, გამოყენებული უნიფიკატორები, აქსიომები და სხვ.) და აჩვენებს მას. გრაფიკულ ინტერფეისს აქვს საბაზისო ფუნქციები, თუმცა სამომავლოდ იგეგმება მისი გაუმჯობესება.

გრაფიკულ ინტერფეისში (და აგრეთვე SQL პროცედურაში) სეკვენციების შესატანად ჩვენ ვიყენებთ LaTeX-ის სტანდარტულ სინტაქსს ლოგიკური ოპერატორებისათვის: `\neg`, `\land`, `\lor`, `\forall`, `\exists` და `\seq` სეკვენციის ნიშნისათვის. სეკვენციების ჩასაწერად ჩვენ გვაქვს შემდეგი წესი: ოპერატორების შემდეგ გამოვიყენოთ ფრჩხილები. ეს წესი განპირობებულია SQL ენის მიერ სტრიქონების დამუშავების სპეციფიკით და არ არის მნიშვნელოვანი, სამომავლოდ იგეგმება მისი მოშორება.

**მაგალითი 3.1.1.** სეკვენცია $\forall x(\neg P(x) \vee Q(x)) \vdash \forall x \exists y(\neg P(x) \vee Q(y))$ ჩვენს სინტაქსში ჩაიწერება შემდეგი სახით:

```
\forall(x,\neg(P(x))\lor(Q(x)))
```

```
\seq(\forall(x,\exists(y,\neg(P(x))\lor(Q(y)))))
```

როდესაც სეკვენცია გადაეწოდება SQL პროცედურებს, გვაქვს სამი ძირითადი პროცედურა: CHECK, APPLY, და AXIOMS რომლებიც ეშვება შესაბამისი თანმიმდევრობით. პირველად, CHECK პროცედურა ამოწმებს მოცემული სეკვენციის სინტაქსს, ანულებს ყველა ცხრილს, კითხულობს სეკვენციის სტრუქტურას სტრიქონიდან და ინახავს შესაბამის ცხრილში. აღნიშნული პროცედურის კოდის მნიშვნელოვანი ნაწილი მოცემულია ქვემოთ:

```
DELETE TREE
WHILE @COUNTER<@LEN BEGIN
  SET @COUNTER=@COUNTER+1
  SET @T=SUBSTRING(@STR_IN,@COUNTER,1)
  IF @T IN (' ','(',')',',') BEGIN
    IF @T=',' BEGIN
      IF SUBSTRING(@TT,1,1) IN ('x','y','z') BEGIN
        SET @V='V'
      END
      ELSE BEGIN
        IF SUBSTRING(@TT,1,1) in ('U','V') BEGIN
          SET @V='U'
        END
        ELSE BEGIN
          SET @V='T'
        END
      END
      EXEC @PARENT_REF=addToTree @ID,@N,@V,@PN,@TT
      SET @N=@N+1
    END
    IF @T=' ' BEGIN
```

```
      SET @N=@N+1
    END
    IF @T='(' BEGIN
      SET @N=@N+1
      EXEC @PARENT_REF=addToTree @ID,@N,'F',@PN,@TT
      SET @PN=@PARENT_REF-@ID
      SET @N=@N+1
    END
    IF @T=')' BEGIN
      IF SUBSTRING(@TT,1,1) IN ('x','y','z') BEGIN
        SET @V='V'
      END
      ELSE BEGIN
        IF SUBSTRING(@TT,1,1) in ('U','V') BEGIN
          SET @V='U'
        END
        ELSE BEGIN
          SET @V='T'
        END
      END
      EXEC @PARENT_REF=addToTree @ID,@N,@V,@PN,@TT
      SELECT @PN=PREF-@ID FROM TREE WHERE REF=@PN+@ID
      SET @N=@N+1
    END
    SET @TT=''
  END
  ELSE BEGIN
    IF @T<>' ' BEGIN
      SET @TT=@TT+@T
    END
```

```
  END
END
```

შემდეგ, პროცედურა APPLY შლის ფორმულას LKU კალკულუსის წესების შესაბამისად, სანამ აქსიომები არ მიიღწევა. გამოყენებული წესების მიმდევრობა ინახება ცალკე ცხრილში. აქაც, წინა ალგორითმის მსგავსად, პრიორიტეტი ენიჭებათ უნარულ წესებს, ვიყენებთ მათ სანამდეც შესაძლებელია და ბინარული წესების გამოყენება ხდება ამის შემდგომ. აღნიშნული პროცედურის კოდის მნიშვნელოვანი ნაწილი მოცემულია ქვემოთ:

```
SELECT @SIDE=PREF % 1000000, @ID=PREF-(PREF % 1000000)
  FROM TREE WHERE REF=@PREF

IF @STR='\forall' BEGIN
  SET @EV=dbo.eigenvariable()
  SELECT @REFA=REF, @VSTR=STR FROM TREE
    WHERE PREF=@PREF AND P='V'
  SELECT @REFB=REF FROM TREE WHERE PREF=@PREF AND P='F'
  IF @SIDE=2 BEGIN
    EXEC SUBSTITUTE @ID, @PREF, @VSTR, @EV
    DELETE TREE WHERE REF IN (@REFA,@PREF)
    UPDATE TREE SET PREF=@ID+@SIDE WHERE REF=@REFB
  END
  ELSE BEGIN
    UPDATE TREE SET STATUS=2 WHERE REF=@PREF
    UPDATE TREE SET STATUS=2
      WHERE dbo.checkRefs(@ID,@PREF,REF)=1
    SELECT @REF1= MAX(REF)-@ID FROM TREE
```

```
      WHERE REF>@ID AND REF<@ID+1000000
    SET @REF2= @REFB+@REF1
    SELECT @STRB=STR FROM TREE WHERE REF=@REFB
    INSERT INTO TREE VALUES(0,@REF2,'F',@ID+@SIDE,@STRB)
    INSERT INTO TREE
      SELECT 0, REF+@REF1,P,PREF+@REF1,STR FROM TREE
        WHERE  dbo.checkRefs(@ID,@REFB,REF)=1
    EXEC SUBSTITUTE @ID, @REF2, @VSTR, @EV
    UPDATE TREE SET PREF=@ID+@SIDE WHERE REF=@REF2
  END
END

IF @STR='\exists' BEGIN
  SET @EV=dbo.eigenvariable()
  SELECT @REFA=REF, @VSTR=STR FROM TREE
    WHERE PREF=@PREF AND P='V'
  SELECT @REFB=REF FROM TREE WHERE PREF=@PREF AND P='F'
  IF @SIDE=1 BEGIN
    EXEC SUBSTITUTE @ID, @PREF, @VSTR, @EV
    DELETE TREE WHERE REF IN (@REFA,@PREF)
    UPDATE TREE SET PREF=@ID+@SIDE WHERE REF=@REFB
  END
  ELSE BEGIN
    UPDATE TREE SET STATUS=2 WHERE REF=@PREF
    UPDATE TREE SET STATUS=2
      WHERE dbo.checkRefs(@ID,@PREF,REF)=1
    SELECT @REF1= MAX(REF)-@ID FROM TREE
      WHERE REF>@ID and REF<@ID+1000000
    SET @REF2= @REFB+@REF1
    SELECT @STRB=STR FROM TREE WHERE REF=@REFB
```

```
    INSERT INTO TREE VALUES(0,@REF2,'F',@ID+@SIDE,@STRB)
    INSERT INTO TREE
      SELECT 0,REF+@REF1,P,PREF+@REF1,STR FROM TREE
        WHERE dbo.checkRefs(@ID,@REFB,REF)=1
    EXEC SUBSTITUTE @ID, @REF2, @VSTR, @EV
    UPDATE TREE SET PREF=@ID+@SIDE WHERE REF=@REF2
  END
END

IF @STR='\neg' BEGIN
  IF @SIDE=1 BEGIN
    SET @SIDE=2
  END
  ELSE BEGIN
    SET @SIDE=1
  END
  SELECT @REF=REF FROM TREE WHERE PREF=@PREF
  DELETE TREE WHERE REF=@PREF
  UPDATE TREE SET PREF=@ID+@SIDE WHERE REF=@REF
END

IF @STR='\land' BEGIN
  IF @SIDE=1 BEGIN
    DELETE TREE WHERE REF=@PREF
    UPDATE TREE SET PREF=@ID+@SIDE WHERE PREF=@PREF
  END
  ELSE BEGIN
    SELECT @NEWSEQID=MAX(REF)+1000000 FROM TREE WHERE PREF=0
    INSERT INTO TREE VALUES(0,@NEWSEQID,'S',0,'\seq')
    SELECT @REFB=REF FROM TREE WHERE PREF=@PREF
```

```
    SELECT @REFA=REF FROM TREE WHERE PREF=@PREF AND REF<>@REFB
    DELETE TREE WHERE REF=@PREF
    UPDATE TREE SET PREF=@ID+@SIDE WHERE REF IN (@REFA,@REFB)
    INSERT INTO TREE
      SELECT STATUS,(REF%1000000)+@NEWSEQID,P,
        (PREF%1000000)+@NEWSEQID,STR FROM TREE
        WHERE STR<>'\seq' AND REF/@ID=1
  END
  SET @REFB=(@REFB%1000000)+@NEWSEQID
  DELETE TREE WHERE dbo.checkRefs(@ID,@REFA,REF)=1
  DELETE TREE WHERE dbo.checkRefs(@NEWSEQID,@REFB,REF)=1
  DELETE TREE WHERE REF IN (@REFA,@REFB)
END

IF @STR='\lor' BEGIN
  IF @SIDE=2 BEGIN
    DELETE TREE WHERE REF=@PREF
    UPDATE TREE SET PREF=@ID+@SIDE WHERE PREF=@PREF
  END
  ELSE BEGIN
    SELECT @NEWSEQID=MAX(REF)+1000000 FROM TREE WHERE PREF=0
    INSERT INTO TREE VALUES(0, @NEWSEQID,'S',0,'\seq')
    SELECT @REFB=REF FROM TREE WHERE PREF=@PREF
    SELECT @REFA=REF FROM TREE WHERE PREF=@PREF AND REF<>@REFB
    DELETE TREE WHERE REF=@PREF
    UPDATE TREE SET PREF=@ID+@SIDE WHERE REF IN (@REFA,@REFB)
    INSERT INTO TREE
      SELECT STATUS,(REF%1000000)+@NEWSEQID,P,
        (PREF%1000000)+@NEWSEQID,STR FROM TREE
        WHERE str<>'\seq' AND REF/@ID=1
```

```
  END
  SET @REFB=(@REFB%1000000)+@NEWSEQID
  DELETE TREE WHERE dbo.checkRefs(@ID,@REFA,REF)=1
  DELETE TREE WHERE dbo.checkRefs(@NEWSEQID,@REFB,REF)=1
  DELETE TREE WHERE REF IN (@REFA,@REFB)
END

IF @STR IS NOT NULL BEGIN
  INSERT INTO RULES VALUES (@STR,@SIDE)
END
```

კვანტორებისათვის, წინა ალგორითმის მსგავსად, ვიყენებთ [AMS98]-ში აღწერილის მსგავს მეთოდს, იმ განსხვავებით, რომ ჩვენ ვაკეთებთ ურანგო უნიფიკაციას. ურანგო უნიფიკაციის პრობლემა, რომელიც შეიცავს მიმდევრობით ცვლადებს საზოგადოდ არა-დასრულებადია. ჩვენს სისტემაში რეალიზებულია ურანგო უნიფიკაციის დასრულებადი, [KM12]-ში აღწერილის მსგავსი ფრაგმენტი. აღნიშნული პროცედურის კოდის მნიშვნელოვანი ნაწილი მოცემულია ქვემოთ:

```
DELETE UTBL
TRUNCATE TABLE UTBLPAIR
DELETE UUNIFIER
SET @N=0
SET @N1=0
SET @ARG=1
Declare BuildTerm1 cursor for
  SELECT PREF,REF,P,STR FROM TREE WHERE PREF=@FPREF
Open BuildTerm1
fetch Next from BuildTerm1 Into @PREF,@REF,@P,@STR
```

```
WHILE @@fetch_status=0 BEGIN
  IF @P='U' BEGIN
    SET @N=@N+1
    SET @N1=0
  END
  ELSE BEGIN
    SET @N1=@N1+1
  END
  INSERT INTO UTBL VALUES(@PREF,@N,@N1,@ARG,@REF,@P,@STR)
  SET @ARG=@ARG+1
  fetch Next from BuildTerm1 Into @PREF,@REF,@P,@STR
END
close BuildTerm1
deallocate BuildTerm1
SET @N=0
SET @N1=0
SET @ARG=1
Declare BuildTerm2 cursor for
  SELECT PREF,REF,P,STR FROM TREE WHERE PREF=@SPREF
Open BuildTerm2
fetch Next from BuildTerm2 Into @PREF,@REF,@P,@STR
WHILE @@fetch_status = 0 BEGIN
  IF @P='U' BEGIN
    SET @N=@N+1
    SET @N1=0
  END
  ELSE BEGIN
    SET @N1=@N1+1
  END
  INSERT INTO UTBL VALUES(@PREF,@N,@N1,@ARG,@REF,@P,@STR)
```

```
  SET @ARG=@ARG+1
  fetch Next from BuildTerm2 Into @PREF,@REF,@P,@STR
END
close BuildTerm2
deallocate BuildTerm2
SET @FARG=1
SET @SARG=1
SET @ALT=0
SELECT @MFARG=MAX(ARG) FROM UTBL WHERE PREF=@FPREF
SELECT @MSARG=MAX(ARG) FROM UTBL WHERE PREF=@SPREF
SET @PP=1
SET @SARGC=@SARG
SET @FARGC=@FARG
WHILE @FARG<=@MFARG BEGIN
  SET @SARG=1
  WHILE @SARG<=@MSARG BEGIN
    SET @SARGC=1
    SET @FARGC=@FARG
    WHILE @FARGC<=@MFARG AND @SARGC<=@MSARG BEGIN
      SET @SPP=1
      SET @FPP=1
      SELECT @FSTR=STR, @FTYPE=TYPE, @FREF=REF FROM UTBL
        WHERE PREF=@FPREF AND ARG=@FARGC
      SELECT @SSTR=STR, @STYPE=TYPE, @SREF=REF FROM UTBL
        WHERE PREF=@SPREF AND ARG=@SARGC
      IF 'U' NOT IN (@FTYPE,@STYPE) BEGIN
        EXEC COMPARER @FPREF,@FARGC,@SPREF,@SARGC, @PP OUTPUT
        IF @PP=0 BEGIN
          SET @FARGC=@MFARG+1
          SET @SARGC=@MSARG+1
```

```
        END
        SET @FPP=@PP
        SET @SPP=@PP
      END
      ELSE BEGIN
        IF @FTYPE='U' BEGIN
          SET @SPP=1
          SET @FPP=0
        END
        ELSE BEGIN
          IF @STYPE='U' BEGIN
            SET @SPP=0
            SET @FPP=1
          END
        END
      END
      IF 'U' IN (@FTYPE,@STYPE) BEGIN
        INSERT INTO UTBLPAIR VALUES(@FPREF,@FREF,@FTYPE,
            @FARGC,@FSTR,@SPREF,@SREF,@STYPE,@SARGC,@SSTR)
      END
      SET @SARGC=@SARGC+@SPP
      SET @FARGC=@FARGC+@FPP
    END
    SET @SARG=@SARG+1
  END
  SET @FARG=@FARG+1
END
```

საბოლოოდ, AXIOMS პროცედურა ამოწმებს მიღებული აქსიომების სისწორეს უნიფიკაციის გამოყენებით და ასრულებს დამტკიცებას შესაბამისი

ჩასმის განხორციელებით. აღნიშნული პროცედურის კოდის მნიშვნელოვანი ნაწილი მოცემულია ქვემოთ:

```
SELECT @isUNRANKED=COUNT(*) FROM TREE WHERE P='U'
SELECT @CNT=COUNT(*) FROM TREE WHERE STR='\seq'
set @CNT1=0
declare GoThrough cursor for
  SELECT REF FROM TREE WHERE STR='\seq'
Open GoThrough
fetch Next from GoThrough Into @REF
WHILE @CNT1<@CNT BEGIN
  declare Axioms cursor for
    SELECT FIRST.REF,FIRST.STR,FIRST.P,
        SECOND.REF,SECOND.STR,SECOND.P
      FROM TREE FIRST,TREE SECOND
      WHERE FIRST.PREF=@REF+1 AND SECOND.PREF=@REF+2
        AND FIRST.STATUS=0 AND SECOND.STATUS=0
  Open Axioms
  fetch Next from Axioms Into
    @REF1,@STR1,@TYPE1,@REF2,@STR2,@TYPE2
  WHILE @@fetch_status=0 BEGIN
    IF @isUNRANKED>0 OR @isUNRANKED is NULL BEGIN
      EXEC UNRANKING @REF1, @REF2
      SELECT @AXIOMS=COUNT(*) FROM RULES
        WHERE OP='AXIOM' AND SIDE=@REF/1000000
      IF @AXIOMS=0 OR @AXIOMS is NULL BEGIN
        INSERT INTO RULES VALUES('AXIOM',@REF/1000000)
      END
    END
    ELSE BEGIN
```

```
      EXEC @P=checkEqualityProc @REF,@REF1,@STR1,@TYPE1,
                                @REF2,@STR2,@TYPE2
      IF @P='1' BEGIN
        UPDATE TREE SET STATUS=1
          WHERE (REF/1000000)=(@REF/1000000)
        INSERT INTO RULES VALUES('AXIOM',@REF/1000000)
      END
    END
    SET @REF11=@REF1
    SET @REF22=@REF2
    fetch Next from Axioms Into
      @REF1,@STR1,@TYPE1,@REF2,@STR2,@TYPE2
  END
  close Axioms
  deallocate Axioms
  set @CNT1=@CNT1+1
  fetch Next from GoThrough Into @REF
END
close GoThrough
deallocate GoThrough
INSERT INTO UNIFIER
  SELECT DISTINCT SREF,SSTR,dbo.newRef(SREF),dbo.newVal(SREF)
    FROM UTBLPAIR
INSERT INTO UNIFIER
  SELECT FREF,FSTR,FREF,'[]' FROM UTBLPAIR WHERE FTYPE='U'
    AND FSTR NOT IN (SELECT VAR1 FROM UNIFIER)
  UNION
  SELECT SREF,SSTR,SREF,'[]' FROM UTBLPAIR WHERE STYPE='U'
    AND SSTR NOT IN (SELECT VAR1 FROM UNIFIER)
```

## 3.2 ტაბლო კალკულუსი ურანგო ლოგიკისათვის

ტაბლოზე დაფუძნებული მსჯელობის მეთოდები ხდება პოპულალური სემანტიკური ვების განვითარებასთან ერთად. ყველა ძირითადი Description Logic ამომხსნელი (მაგ. Racer [HM01a], FaCT++ [TH06], Pellet [SPG+07], etc.) იყენებს ტაბლო კალკულუსს როგორც მთავარი მსჯელობის მეთოდს (იხილეთ: [HV06, BP10]).

ტაბლო კალკულუსი [Smu68] დაფუძნებულია უარყოფის პრინციპზე. როდესაც გვაქვს მოცემული ფორმულა, ვუშვებთ მის უარყოფას და გარკვეული წესების გამოყენებით ვშლით ქვეფორმულებად. დაშლის პროცედურა გაქვს ფორმულათა ხეს. თუ ყოველი ხის ტოტი დაიხურება (ტოტი შეიცავს საწინააღმდეგო ფორმულებს), გამოვა რომ მოცემული ფორმულა არ არის შესრულებადი. ტაბლო მეთოდის უპირატესობა სხვა სისტემებთან შედარებით არის ის, რომ მას შეუძლია მოდელის აგება შესრულებადი ფორმულისათვის და საწინააღმდეგო მოდელის პოვნა არა-შესრულებადი ფორმულისათვის.

ლიტერატურაში არსებობს ტაბლო კალკულუსის მრავალი გაუმჯობესება და მოადიფიკაცია (იხილეთ, მაგ.: [RV01a, DGHP13]). ამაში მოიაზრება ტაბლო ინტუიციური, დროებითი, მოდალური ქვესტრუქტურული, არამონოტონური, მრავალმნიშვნელობიანი და სხვა ლოგიკებისათვის. ჩვენს ხელთ არსებული ინფორმაციით, ჯერ არ არსებობს ტაბლო კალკულუსი ურანგო ლოგიკისათვის, ის შემოღებულ იქნა ავტორის მიერ თანაავტორებთან ერთად [DKR17]-ში.

წარმოგიდგენთ კლასიკური პირველი რიგის ტაბლო კალკულუსს ურანგო ლოგიკისათვის. კალკულუსში წესების შესამცირებლად ჩვენ გავიხილავთ ფორმულებს მხოლოდ *უარყოფათა ნორმალურ ფორმაში* (NNF). კლასიკური პირველი რიგის სემანტიკა განსაზღვრულია ჩვენი ენისთვის,რომელიც უფლებას გვაძლევს ფორმულის სკოლემიზაციის და ტრანსფორმაციის NNF-ში სტანდარტული გზით.

ურანგო ტაბლო კალკულუსი ფორმულებისთვის უარყოფათა ნორმალურ

ფორმაში შედგება შემდეგი წესებისაგან :

$$\frac{A \wedge B}{\begin{array}{c}A\\B\end{array}}\wedge \qquad \frac{A \vee B}{A \quad B}\vee \qquad \frac{\forall x A}{A[x \mapsto t]}\forall_i \qquad \frac{\forall \overline{x} A}{A[\overline{x} \mapsto \overline{s}]}\forall_s$$

**შენიშვნა 3.2.1.** განვსაზღრავთ წესებს შორის განსხვავებებს:

- $\wedge$-წესი გვიჩვენებს ერთი ან ორივე ოპერატორის გაშლას ერთიდაიმეავე ტოტში. განსხვავებით $\vee$-წესისგან, რომელიც ქმნის ალტერნატიულ ტოტს თითოეული ოპერატორისთვის.
- $\forall_i$ კვანტორულ წესში ინდივიდუალური ცვლადი იცვლება ინდივიდუალური ტერმით, ხოლო $\forall_s$-წესის შემთხვევაში მიმდევრობითი ცვლადი ჩანაცვლდება ძირითადი მიმდევრობითი ტერმით, მათ შორის ცარიელითაც.

ტაბლოს ტოტი *დახურულია* თუ ის შეიცავს ორივეს, ფორმულას და მის მოპირდაპირე ფორმულას; სხვა შემთხვევაში ტოტს ეწოდება *ღია*. ტაბლო დახურულად ითვლება თუ მისი ყველა ტოტი დახურულია.

ჩვენ ვიყენებთ ურანგო უნიფიკაციას ტაბლო კალკულუსში, როგორც მექანიზმს რომელიც წყვიტავს დახურულია თუ არა ტოტი. ის არჩევს ტერმების ჩანაცვლებისთვის კვანტორულ წესში. როგორც ზემოთ აღვნიშნეთ, ურანგო უნიფიკაცია არ არის სასრული, ამან შეიძლება გამოიწვიოს ალგორითმის არა-დასრულებადობა. მიუხედავად ამისა, პრაქტიკაში შესაძლებელია მოვახდინოთ უნიფიკაციის შეზღუდვა შეთანადების პრობლემაზე. ეს წარმოდგენილია შემდეგ სექციაში.

ტაბლო კალკულუსი არის კონფლუენტური. სხვა სიტყვებით, წესის უკან დაბრუნება საჭირო არ არის. ერთადერთი "უკან დაბრუნების" წერტილი არის ჩანაცვლებები $\forall_i$ და $\forall_s$ კვანტორულ წესებში.

რომ შევამოწმოთ ფორმულა $A$ სწორია თუ არა, უნდა დავუშვათ მისი საწინაღმდეგო $\neg A$ რომელიც გარდაიქმნება NNF-ში და შემდეგ დაიშლება ტაბლოს წესებით. თუ ტაბლოს ყველა ტოტი დაიხურება, ნიშნავს რომ $A$ ზოგადმართებულია. შევნიშნოთ რომ, თუ $B$ არის $A_1, \ldots, A_n$ ფორმულების

ლოგიკური შედეგი, მაშინ $(A_1 \wedge \cdots \wedge A_n) \to B$ ფორმულის ზოგადმართებულობა უნდა შემოწმდეს.

საბოლოოდ, თუ ტაბლო არ იხურება ნიშნავს, რომ ფორმულა შესრულებადია და (კონტრ) მოდელის წაიკითხვა შესაძლებელია ღია ტოტებიდან.

**მაგალითი 3.2.2.** განვიხილოთ ფორმულა $A$: $\neg\,(P(\bar{y}, a) \wedge (\forall \bar{x})(\neg P(\bar{x}) \vee P(f(\bar{x}))) \vee P(f(a, \bar{y}))$ და ავაგოთ მისი დამტკიცება ტაბლო მეთოდის გამოყენებით. ამისათვის უნდა გამოვთვალოთ $\neg A$ და დავიწყოთ მისი დაშლა ტაბლოს მოცემული წესების შესაბამისად. $\neg A$: $P(\bar{y}, a) \wedge (\forall \bar{x})(\neg P(\bar{x}) \vee P(f(\bar{x}))) \wedge \neg P(f(a, \bar{y}))$, მაშინ გვექნება შემდეგი ხე:

$$\dfrac{\dfrac{\dfrac{P(\bar{y}, a) \wedge (\forall \bar{x})(\neg P(\bar{x}) \vee P(f(\bar{x}))) \wedge \neg P(f(a, \bar{y}))}{P(\bar{y}, a)}\wedge}{\dfrac{(\forall \bar{x})(\neg P(\bar{x}) \vee P(f(\bar{x})))}{\neg P(\bar{x}) \vee P(f(\bar{x}))}\forall_s}}{\begin{array}{cc}\neg P(\bar{x}) & P(f(\bar{x}))\end{array}\vee}$$
$$\neg P(f(a, \bar{y}))$$

ამ ხის დასახურად საჭიროა ჯერ $P(\bar{y}, a)$ და $P(\bar{x})$ წყვილის უნიფიკაცია, რომელიც დასრულებადია და გვაძლევს უნიფიკატორს $\sigma = [\bar{x} \mapsto \ulcorner \bar{y}, a \urcorner]$. შესაბამისად მივიღებთ:

$$\dfrac{\dfrac{\dfrac{P(\bar{y}, a) \wedge (\forall \bar{x})(\neg P(\bar{x}) \vee P(f(\bar{x}))) \wedge \neg P(f(a, \bar{y}))}{P(\bar{y}, a)}\wedge}{\dfrac{(\forall \bar{x})(\neg P(\bar{x}) \vee P(f(\bar{x})))}{\neg P(\bar{y}, a) \vee P(f(\bar{y}, a))}\forall_s}}{\begin{array}{cc}\neg P(\bar{y}, a) & P(f(\bar{y}, a)) \\ \times & \neg P(f(a, \bar{y}))\end{array}\vee}$$

შემდეგ გვჭირდება $P(f(\bar{x}))\sigma = P(f(\bar{y}, a))$ და $P(f(a, \bar{y}))$ წყვილის უნიფიკაცია. ამ შემთხვევაში უნიფიკაცია დასრულებადი არ არის. გვაქვს უნიფიკატორების უსასრულო მიმდევრობა $[\bar{y} \mapsto \ulcorner \urcorner]$, $[\bar{y} \mapsto \ulcorner a \urcorner]$, $[\bar{y} \mapsto \ulcorner a, a \urcorner], \ldots$. ავიღოთ ნებისმიერი უნიფიკატორი ამ სიმრავლიდან, ვთქვათ $\theta = [\bar{y} \mapsto \ulcorner a \urcorner]$, მაშინ დახურულ ტაბლო ხეს ექნება სახე:

$$\dfrac{\dfrac{\dfrac{\dfrac{P(a,a) \wedge (\forall \bar{x})(\neg P(\bar{x}) \vee P(f(\bar{x}))) \wedge \neg P(f(a,a))}{P(a,a)}\wedge}{(\forall \bar{x})(\neg P(\bar{x}) \vee P(f(\bar{x})))}\forall_s}{\neg P(a,a) \vee P(f(a,a))}\vee}{\begin{array}{cc}\neg P(a,a) & P(f(a,a)) \\ \times & \neg P(f(a,a)) \\ & \times\end{array}}$$

**წინადადება 3.2.1.** *ურანგო ტაბლო კალკულუსი არის სწორი.*

*დამტკიცება.* მარტივი შესამოწმებელია, რომ თითოეული წესისთვის ზედა ფორმულიდან გამომდინარეობს ქვედა ფორმულა. □

**წინადადება 3.2.2.** *ურანგო ტაბლო კალკულუსი არის სრული.*

*დამტკიცება.* სისრულის დამტკიცება ხორციელდება ანალოგიური გზით, რაც მოცემულია სეკვენციათა კალკულუსისათვის [KB04]-ში. □

### 3.2.1 გამოყენებები სემანტიკურ ქსელში

მსჯელობა მნიშვნელოვან როლს თამაშობს ინტერნეტ სივრცეში მონაცემთა დამუშავების პროცესში და ხდის უფრო "ინტელექტუალურს". სემანტიკური ვები ამატებს მეტამონაცემებს ვებ-რესურსებისთვის, რომელიც შეიძლება გამოყენებული იქნას როგორც მანქანებისათვის წაკითხვადი ინფორმაცია დამატებითი მსჯელობის მექანიზმის შესაქმნელად. ორივე, მონაცემებისა და მეტა მონაცემების შეკითხვაზე ფორმალიზებებს უნდა ჰქონდეთ გარკვეული დასაბუთებული პასუხები. მონაცემიც და მეტამონაცემიც არის ტექსტური ინფორმაცია, რომელიც შეიძლება გამოვითვალოთ შემოსაზღვრული ტერმების (ცვლადის გარეშე ტერმები) გამოყენებით. კითხვები შეიძლება შეიცავდეს ინდივიდუალურ და მიმდევრობით ცვლადებს. ამდენად ვებთან დაკავშირებული აპლიკაციებში ინფორმაციის მოძიებისათვის, უნიფიკაციის ნაცვლად ჩვენ გვჭირდება შეთანადების პრობლემის ამოხსნა, რომელიც არის გადაწყვეტადი და სასრული (მეტი ინფორმაციისათვის იხილეთ, მაგ.: [KM12]).

ჩვენ ვარჩევთ სახელიან და ანონიმურ ცვლადებს. ანონიმური ინდივიდუალური ცვლადები აღინიშნება $\_i$ და შეიძლება გამოყენებული იქნას

როგორც თავისუფალი ადგილი ნებისმიერი ინდივიდუალური ტერმისთვის, ხოლო ანონიმური მიმდევრობითი ცვლადი აღინიშნება $\_s$ და გამოიყენება როგორც თავისუფალი ადგილი ნებისმიერი მიმდევრობითი ტერმისთვის. შესაბამისად, სახელიანი ინდივიდუალური და სახელიანი მიმდევრობითი ცვლადები გამოიყენება ტერმის ან ნებისმიერი მიმდევრობითი ტერმის შესაცვლელად. ამიტომ ანონიმური ცვლადები ხელს უწყობს დოკუმენტის შეუსაბამო ნაწილის გამოტოვებას, ხოლო სახელიანი ცვლადები ეხმარება მონაცემთა მიმდევრობის ამოხსნას, მისი სიგრძისა და სტრუქტურის ცოდნის გარეშე.

ამ ნაწილში ჩვენ წარმოგიდგენთ საბაზისო მსჯელობის შესაძლებლობებს ურანგო ტაბლო კალკულუსით შემდეგი მაგალითის მეშვეობით "მეგორების წრე" (Clique of Friends) [Sch04a]. ეს მაგალითი გვაჩვენებს ზოგიერთ საბაზისო მსჯელობას სემანტიკური ვებისთვის. არ იყენებს რომელიმე კონკრეტულ სემანტიკურ ვებ-ენას.

განვიხილოთ მისამართების წიგნების სიმრავლე, სადაც ყოველ მისამართების წიგნს ჰყავს თავისი მესაკუთრე და ჩანაწერების სიმრავლე, რომელშიც ზოგიერთი ჩაწერილია როგორც მეგობარი (`friend`), ანუ ამ ჩანაწერთან დაკავშირებული პერსონა განიხილება, როგორც მისამართების წიგნის მესაკუთრის მეგობარი. XML-ში ეს მისამართების წიგნების სიმრავლე წარმოდგენილია პირდაპირი წესით, შემდეგნარიად:

```
<address-books>
   <address-book>
      <owner>Donald Duck</owner>
      <entry>
         <name>Daisy Duck</name>
         <friend/>
      </entry>
      <entry>
         <name>Scrooge McDuck</name>
      </entry>
```

```
   </address-book>
   <address-book>
      <owner>Daisy Duck</owner>
      <entry>
         <name>Gladstone Duck</name>
         <friend/>
      </entry>
      <entry>
         <name>Ratchet Gearloose</name>
         <friend/>
      </entry>
   </address-book>
</address-books>
```

სიმრავლე შეიცავს ორ მისამართების წიგნს, პირველის მესაკუთრე არის დონალდ დაკი (Donald Duck) და მეორესი კი დეიზი დაკი (Daisy Duck). დონალდის მისამართების წიგნს აქვს ორი ჩანაწერი, ერთ არის სკრუჩი (Scrooge), ხოლო მეორე კი დეიზი (Daisy), და მხოლოდ დეიზი (Daisy) აღნიშნულია როგორც მეგობარი (friend). დეიზის მისამართების წიგნს აქვს ასევე ორი ჩანაწერი და ორივე აღნიშნულია როგორც მეგობარი (friend).

დონალდის "მეგობრების წრე" (*clique-of-friends*) წარმოადგენს ყველა პრესონის სიმრავლეს სადაც თითოეული არის დონალდის პირდაპირი მეგობარი (ასეთი არის მხოლოდ დეიზი (Daisy)) ან მეგობრის მეგობარი (Gladstone და Ratchet), ან მეგობრის მეგობრის მეგობარი (ასეთი არ გვყავს ამ მაგალითში), და ა.შ. რომ მოვძებნოთ ეს მეგობრები, ჩვენ უნდა განვსაზღვროთ კავშირი "მეგობრად ყოფნის" ("a friend of") და ეს არის ტრანზიტიული ჩაკეტვა.

ფორმალური სინტაქსიდან მარტივი გადახვევის საფუძველზე გამოვიყენეთ შემდეგი აბრევიატურები:

- ფიქსირებულ ადგილიანი ფუნქციონალური სიმბოლოები

$$f_r^o ::= \texttt{owner}$$

$$f_r^n ::= \texttt{name}$$

$$f_r ::= \texttt{friend}$$

- მოძრავ ადგილიანი ფუნქციონალური სიმბოლოები

$$f_u^{ab} ::= \texttt{address-book}$$

$$f_u^e ::= \texttt{entry}$$

- მოძრავ ადგილიანი პრედიკატული სიმბოლოები

$$p_u^{abs} ::= \texttt{address-books}$$

$$p_u^{fo} ::= \texttt{friend-of}$$

$$p_u^{fof} ::= \texttt{friend-of-friend}$$

როგორც უკვე აღვნიშნეთ XML ცოდნის ბაზაში წარმოდგენილია შემდეგნაირად:

$$\begin{aligned}\texttt{XML} \equiv & p_u^{abs}(f_u^{ab}(f_r^o(\texttt{DonaldDuck}),\\ & \quad f_u^e(f_r^n(\texttt{DaisyDuck}), f_r),\\ & \quad f_u^e(f_r^n(\texttt{ScroogeMcDuck}))),\\ & f_u^{ab}(f_r^o(\texttt{DaisyDuck}),\\ & \quad f_u^e(f_r^n(\texttt{GladstoneDuck}), f_r),\\ & \quad f_u^e(f_r^n(\texttt{RatchetGearloose}), f_r)))\end{aligned}$$

ჩვენ განვსაზღვრეთ კავშირები *მეგობრის მეგობარი* და *მეგობრის მეგობარის მეგობარი* ცოდნის ბაზაში შემდეგნაირად:

$$\begin{aligned}p_u^{fo}(\overline{x}, \overline{y}) &\equiv p_u^{abs}(\_s, f_u^{ab}(f_r^o(\overline{x}), \_s, f_u^e(f_r^n(\overline{y}), f_r), \_s), \_s),\\ p_u^{fof}(\overline{x}, \overline{y}) &\equiv p_u^{fo}(\overline{x}, \overline{y}),\\ p_u^{fof}(\overline{x}, \overline{y}) &\equiv p_u^{fo}(\overline{x}, \overline{z}) \wedge p_u^{fof}(\overline{z}, \overline{y}),\end{aligned}$$

სადაც $\_s$ არის ანონიმური მიმდევრობით ცვლადი, რომელიც შეიძლება იყოს ნებისმიერი მიმდევრობითი ტერმი, მათ შორის ცარიელიც.

ისმის კითხვა $KB \rightarrow \exists\overline{x}\ p_u^{fof}(DonaldDuck, \overline{x})$. შევნიშნოთ რომ $p_u^{fof}$ პრედიკატი შეიძლება წარმოვადგინოთ როგორც ფორმულა $p_u^{fo}(\overline{x}, \overline{y}) \vee (\neg p_u^{fo}(\overline{x}, \overline{y}) \wedge p_u^{fo}(\overline{x}, \overline{z}) \wedge p_u^{fof}(\overline{z}, \overline{y}))$. შემდეგ, კითხვის უარყოფა (NNF-ში გარდაქმნის შემდეგ) გვექნება:

$$KB \wedge \forall\overline{x}\ \neg p_u^{fo}(DonaldDuck, \overline{x}) \\ \wedge \forall\overline{x}\ (p_u^{fo}(DonaldDuck, \overline{x}) \vee \neg p_u^{fo}(DonaldDuck, \overline{y}) \vee \neg p_u^{fof}(\overline{y}, \overline{x}))$$

უარყოფა შეიცავს შემდეგ ნაბიჯებს:

$$\dfrac{\dfrac{KB \wedge \forall\overline{x}\ \neg p_u^{fo}(DonaldDuck, \overline{x}) \wedge ...}{KB}\wedge}{\dfrac{\forall\overline{x}\ \neg p_u^{fo}(DonaldDuck, \overline{x})}{\neg p_u^{fo}(DonaldDuck, \overline{x})\sigma}\forall_s}\times$$

სადაც $\sigma = \{\overline{x} \mapsto \texttt{DaisyDuck}\}$. თუ გვინდა რომ ყველა შედეგი მოვიძიოთ, მშინ ჩვენ უნდა განვაგრძოთ დაშლა მეორე ფორმულის:

$$\dfrac{\dfrac{KB \wedge \forall\overline{x}\ \neg p_u^{fo}(DonaldDuck, \overline{x}) \wedge ...}{KB}\wedge}{\begin{array}{c}\forall\overline{x}\ (p_u^{fo}(DonaldDuck, \overline{x}) \vee \neg p_u^{fo}(DonaldDuck, \overline{y}) \vee \neg p_u^{fof}(\overline{y}, \overline{x})) \\ \vdots \\ \dfrac{(\neg p_u^{fo}(DonaldDuck, \overline{y}) \vee \neg p_u^{fof}(\overline{y}, \overline{x}))\theta}{\neg p_u^{fo}(DonaldDuck, \overline{y})\theta \quad \neg p_u^{fof}(\overline{y}, \overline{x})\theta}\vee\end{array}}\begin{array}{c}\times \\ \times\end{array}$$

საადაც $\theta$ არის თითოული ჩასმა:

$$\{\overline{x} \mapsto \texttt{GladstoneDuck}, \overline{y} \mapsto \texttt{DaisyDuck}\}$$

$$\{\overline{x} \mapsto \texttt{RatchetGearloose}, \overline{y} \mapsto \texttt{DaisyDuck}\}$$

ამ ჩასმებიდან გამომდინარე ჩვენ შეგვიძლია წავიკითხოთ კითხვაზე პასუხი:

$$\{\texttt{DaisyDuck}, \texttt{GladstoneDuck}, \texttt{RatchetGearloose}\}$$

გაითვალისწინეთ, რომ კალკულუსს შეუძლია არამხოლოდ ფორმულის დამტკიცება, ასევე შესაძლებელია გამოთვლა მიმდევრობითი და ინდივიდუალური ცვლადების, რომელიც გვეხმარება ფორმულის დამტკიცებაში.

### 3.2.2 ურანგო სკოლემიზაცია

საწინააღმდეგოს დაშვებაზე დაფუძნებულ კალკულუსებში სკოლემიზაცია არის პროცესი, როდესაც ფორმულებიდან ვაშორებთ არსებობის კვანტორებს. არსებობს სკოლემიზაციის განსაზღვრისთვის სხვადასხვა გზები:

**პრენექსი:** ტრადიციული გზა, რომ მივიღოთ ფორმულისთვის სკოლემის ნორმალური ფორმა, პირველად ფორმულა უნდა გარდავქმნათ პრენექსის ნორმალურ ფორმაში და შემდეგ ჩამოვაშოროთ არსებობის კვანტორები, რაც შესაძლებელია შესაბამისი დაბმული ცვლადის ჩანაცვლებით, ახალი $n$-ური ფუნქციონალური სიმბოლოების მეშვეობთ, $n \geq 0$, სადაც $n$ არის უნივერსალური კვანტორების რიცხვი, რომლის მოქმედების არეშიც ხვდება არსებობის კვანტორი.

**სტრუქტურული:** ამ მეთოდს არ სჭირდება პრენექსის ნორმალურ ფორმაში გარდაქმნა. ეს მეთოდი უფრო ზოგადია, რადგან შესაძლებელია ძლიერი კვანტორების მოშორება ფორმულიდან. წესი მსგავსია – ძლიერი კვანტორი $(Qx)$ დამოკიდებულია სუსტ კვანტორებზე, რომელთა მოქმედების არეშიც ხვდება $(Qx)$. ამავე გზით შესაძლებელია სუსტი კვანტორების მოშორება, მაგრამ ამ პროცესს ლიტერატურაში უწოდებენ *ჰერბრენდიზაციას* (იხილეთ, მაგ.: [Bus95, Bus98]).

**ანტიპრენექსი:** ეს მეთოდი სტრუქტრული სკოლემიზაციის მსგავსია, მაგრამ პრენექსის ნორმალურ ფორმისგან განსხვავებით, კვანტორები შედის ფორმულის შიგნით, კვანტორების გადატანის წესების საშუალებით, რომ შევამციროთ კვანტორების მოქმედების არე. საზოგადოდ ამ პროცესს მივყავართ უფრო პატარა სკოლემის თერმებამდე.

ქვემოთ განვიხილავთ სტრუქტურულ სკოლემიზაციას, რომელიც უფრო ზოგადია და პრენექსისგან და ანტიპრენექსისგან განსხვავებით არ მოითხოვს რაიმე წინამორბედ ნაბიჯებს.

შემდეგი განსაზღვრებები განმარტავენ ზოგიერთ უკვე ნახსენებ ტერმინს, რომელთა ფორმალური განმარტებაც დაგვჭირდება შედეგების გამოსაყვანად.

**განსაზღვრება 3.2.3** (ძლიერი და სუსტი კვანტორები)**.** კვანტორის პოლარობა განისაზღვრება შემდეგი ინდუქციური გზით:

- $(Qx)$ არის დადებითი $(Qx)A$-ში.
- თუ $(Qx)$ არის დადებითი (უარყოფითი) $A$ ან $B$-ში, მაშინ ის არის დადებითი (უარყოფითი) $A \wedge B$ და $A \vee B$-ში.
- თუ $(Qx)$ არის დადებითი (უარყოფითი) $B$-ში, მაშინ ის არის დადებითი (უარყოფითი) $A \to B$-ში.
- თუ $(Qx)$ არის დადებითი (უარყოფითი) $A$-ში, მაშინ ის არის უარყოფითი (დადებითი) $\neg A$ და $A \to B$-ში.

დადებით უნივერსალურ (არსებობის) კვანტორებს და უარყოფით არსებობის (უნივერსალურ) კვანტორებს ეწოდებათ *ძლიერი* (*სუსტი*) კვანტორები.

**განსაზღვრება 3.2.4** (კვანტორის მოქმედების არე)**.** ვთქვათ $A$ არის ფორმულა და $(Qx)B$ მისი ქვეფორმულა. ჩვენ ვიტყვით, რომ $B$ არის $(Qx)$ კვანტორის მოქმედების არეში და $B$-ს ყოველი ქვეფორმულისათვის $(Q'y)C$, კვანტორი $(Q'y)$ არის $(Qx)$ კვანტორის მოქმედების არეში.

**განსაზღვრება 3.2.5** (კვანტორის გამოტოვება)**.** დავუშვათ $A$ არის ფორმულა. $A_{-(Qx)}$ აღნიშნავს $A$ ფორმულას, სადაც $(Qx)$ კვანტორის შემოსვლა არის გამოტოვებული.

**მაგალითი 3.2.6.** ვთქვათ $A$: $B \wedge (\forall x)(\exists \overline{x}) p_u(x, \overline{x})$. მაშინ $A_{-(\forall x)}$: $B \wedge (\exists \overline{x}) p_u(x, \overline{x})$ და $A_{-(\exists \overline{x})}$: $B \wedge (\forall x) p_u(x, \overline{x})$.

ჩვენ ვიტყვით, რომ ფორმულა არის პრენექსის ფორმაში, თუ მას აქვს შემდეგი ფორმა $(Q_1x_1)\cdots(Q_nx_n)A$, სადაც $A$ არის კვანტორებისაგან თავისუფალი ფორმულა. ყველა ფრომულა შესაძლებელია გარდაიქმნას პრენექსის ფორმაში შემდეგი კვანტორების გადატანის წესების გამოყენებით:[2]

$$\begin{aligned}
\neg(Qx)A &\iff (Q^dx)\neg A\\
(\exists x)(A\vee B) &\iff (\exists x)A\vee(\exists x)B\\
(\forall x)(A\wedge B) &\iff (\forall x)A\wedge(\forall x)B\\
(\exists x)(A\to B) &\iff (\forall x)A\to(\exists x)B
\end{aligned}$$

და თუ $x$ არ არის თავისუფალი $B$-ში, მაშინ

$$\begin{aligned}
(Qx)(A\vee B) &\iff (Qx)A\vee B\\
(Qx)(A\wedge B) &\iff (Qx)A\wedge B\\
(Qx)(A\to B) &\iff (Q^dx)A\to B\\
(Qx)(B\to A) &\iff B\to(Qx)A
\end{aligned}$$

პირველი რიგის ურანგო ფორმულებისთვის განვავრცოთ განმარტებები [BEL01]-დან.

**განსაზღვრება 3.2.7** (სტრუქტურული სკოლემიზაცია)**.** ვთქვათ $A$ არის ურანგო ფორმულა. ჩვენ განვსაზღვრავთ სტრუქტრურული სკოლემიზაციის ოპერატორს $sk()$ შემდეგი გზით:

- თუ $A$ არ შეიცავს ძლიერ კვანტორებს, მაშინ $sk(A)=A$.
- ვთქვათ $(Qx)$ არის პირველი ძლიერი კვანტორი, რომელიც გვხვდება $A$-ში. თუ $(Qx)$ არის სუსტი კვანტორების $(Q_1x_1),\dots,(Q_nx_n)$, $n\geq 0$, მოქმედების არეში, მაშინ ჩვენ გამოვარჩევთ შემდეგ შემთხვევებს:
  - თუ $n>0$ და $x_j\in\mathcal{V}_{\mathsf{S}}$ ნებისმიერი $j=1,\dots,n$, მაშინ ვიღებთ ახალ მოძრავადგილიან ფუნქციონალურ სიმბოლოს $f_u$ რომელიც არ გვხვდება $A$-ში და $sk(A)=sk(A_{-(Qx)}[x\mapsto f_u(x_1,\dots,x_n)])$;

[2] ფორმულის პრენექსის ფორმაში გადასაყვანად წესები გამოიყენება მარჯვნიდან მარცხნივ. მეორე მიმართულება გამოიყენება ფორმულის ანტიპრენექს ფორმაში გადასაყვანად.

  - სხვაგვარად, ავიღოთ ახალი $n$-ური ფუნქციონალური სიმბოლო $f$, რომელიც არ გვხვდება $A$-ში და $sk(A) = sk(A_{-(Qx)}[x \mapsto f(x_1, \dots, x_n)])$; შევნიშნოთ რომ, თუ $n = 0$, მაშინ $f$ არის მუდმივა.

- ვთქვათ $(Q\overline{x})$ არის პირველი ძლიერი კვანტორი რომელიც გვხვდება $A$-ში. თუ $(Q\overline{x})$ არის სუსტი კვანტორების $(Q_1x_1), \dots, (Q_nx_n)$, $n \geq 0$, მოქმედების არეში, მაშინ ჩვენ გამოვარჩევთ შემდეგ შემთხვევებს:

  - თუ $n > 0$ და $x_j \in \mathcal{V}_\mathsf{S}$ ნებისმიერი $j = 1, \dots, n$, მაშინ ვიღებთ ახალ მოძრავადგილიან ფუნქციონალურ სიმბოლოებს $f_u^1, \dots, f_u^m$, $m > 0$, რომლებიც არ გვხვდება $A$-ში და $sk(A) = sk(A_{-(Q\overline{x})}[\overline{x} \mapsto \ulcorner f_u^1(x_1, \dots, x_n), \dots, f_u^m(x_1, \dots, x_n)\urcorner])$;
  - სხვაგვარად, ავიღოთ ახალი $n$-ური ფუნქციონალური სიმბოლოები $f_1, \dots, f_m$, $m > 0$, რომელიც არ გვხვდება $A$-ში და $sk(A) = sk(A_{-(Q\overline{x})}[\overline{x} \mapsto \ulcorner f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)\urcorner])$; შევნიშნოთ, რომ თუ $n = 0$, მაშინ $f_1, \dots, f_m$ არიან მუდმივები.

**შენიშვნა 3.2.8.** გასათვალისწინებელია, რომ თუ განმარტება 3.2.7-ში ჩვენ სიტყვებს "ძლიერი" და "სუსტი" გავუცვლით ადგილებს, მაშინ მივიღებთ ფორმულის ჰერბრენდიზაციას – სკოლემიზაციის შებრუნებულს. აგრეთვე უნდა გავითვალისწინოთ, რომ სუსტი კვატორები ხდებიან ძლიერები როდესაც ხდება ფორმულის ურაყოფა საწინააღმდეგოს დაშვებაზე დაფუძნებულ კალკულუსებში.

**განსაზღვრება 3.2.9.** ვთქვათ $A$ არის ურანგო ფორმულა და $A'$ კი მისი პრენექსის ფორმა; მაშინ $sk(A')$ არის $A$-ს პრენექსის სკოლემის ფორმა. ანალოგიურად ჩვენ განვსაზღვრავთ ანტიპრენექსისის სკოლემის ფორმას $A$-სთვის.

**მაგალითი 3.2.10.** განვიხილოთ ფორმულა:

$$A\colon (\exists x)\,((\forall y)p_u(x,y) \wedge (\forall \overline{x})q_u(\overline{x})) \wedge (\exists \overline{y})\,(p_u(\overline{y}) \vee q_u(\overline{y}))$$

და ავაგოთ მისი სტრუქტურული, ანტი-პრენექსული და პრენექსული სკოლემის ფორმები. განსაზღვრება 3.2.7-ის თანახმად გვექნება:

$$(\exists x)\,(p_u(x, f(x)) \wedge q_u(\ulcorner f_1(x), \dots, f_k(x)\urcorner)) \wedge (\exists \overline{y})\,(p_u(\overline{y}) \vee q_u(\overline{y}))$$

სადაც $f, f_1, \dots, f_k$ არიან ერთმანეთისაგან განსხვავებული ფუნქციონალური სიმბოლოები ფიქსირებული ადგილიანობით 1.

თუ გამოვიყენებთ კვანტორების გადატანის წესებს, მაშინ მივიღებთ $A$ ფორმულის ანტი-პრენექსულ ფორმას:

$$A' : (\exists x)(\forall y)p_u(x, y) \wedge (\forall \overline{x})q_u(\overline{x}) \wedge (\exists \overline{y})\,(p_u(\overline{y}) \vee q_u(\overline{y}))$$

ისევ განსაზღვრება 3.2.7-ის მიხედვით მივიღებთ:

$$(\exists x)p_u(x, f(x)) \wedge q_u(\ulcorner f_1, \dots, f_k \urcorner) \wedge (\exists \overline{y})\,(p_u(\overline{y}) \vee q_u(\overline{y}))$$

სადაც $f_1, \dots, f_k$ არიან ერთმანეთისაგან განსხვავებული კონსტანტები (ფუნქციონალური სიმბოლოები ფიქსირებული ადგილიანობით 0), რომლებიც არ არიან დამოკიდებული $x$ ცვლადზე.

$A$ ფორმულას გააჩნია რამდენიმე პრენექსის ნორმალური ფორმა, სადაც კვანტორების დალაგება შესაძლებელია იყოს შემდეგი თანმიმდევრობებით: $(\exists x)(\forall y)(\forall \overline{x})(\exists \overline{y})$, $(\exists x)(\forall \overline{x})(\forall y)(\exists \overline{y})$, $(\exists \overline{y})(\exists x)(\forall y)(\forall \overline{x})$, ან $(\exists \overline{y})(\exists x)(\forall \overline{x})(\forall y)$. პირველ ორ შემთხვევაში სკოლემის თერმები იქნება სტრუქტურული სკოლემიზაციისას მიღებული ტერმების მსგავსი, ამიტომ საინტერესოა ბოლო ორიდან ერთ-ერთი ვარიანტის განხილვა (რადგან ისინიც ქმნიან ერთმანეთის მსგავს სკოლემის თერმებს), ვთქვათ:

$$A'' : (\exists \overline{y})(\exists x)(\forall y)(\forall \overline{x})\,(p_u(x, y) \wedge q_u(\overline{x}) \wedge (p_u(\overline{y}) \vee q_u(\overline{y})))$$

მაშინ, განსაზღვრება 3.2.7-ის თანახმად გვექნება:

$$(\exists \overline{y})(\exists x)\,(p_u(x, f_u(\overline{y}, x)) \wedge q_u(\ulcorner f_u^1(\overline{y}, x), \dots, f_u^k(\overline{y}, x) \urcorner) \wedge (p_u(\overline{y}) \vee q_u(\overline{y})))$$

სადაც, $f_u, f_u^1, \dots, f_u^k$ არიან ერთმანეთისაგან განსხვავებული ფუნქციონალური სიმბოლოები, რომელთა ადგილიანობა ფიქსირებული არ არის.

**თეორემა 3.2.3** (სისწორე)**.** *თუ $A$ ზოგადმართებულია, მაშინ $sk(A)$ ასევე ზოგადმართებულია.*

*დამტკიცება.* ინდუქციით ძლიერი კვანტორების რაოდენობაზე $A$ ფორმულაში. საბაზისო შემთხვევა ტრივიალურია: $A = sk(A)$.

დავუშვათ, რომ თეორემა მოქმედებს ყველა ფორმულაზე, რომელიც შეიცავს არაუმეტეს $n$ ძლიერ კვანტორს და $A$ შეიცავს $n+1$ ძლიერ კვანტორს. მაშინ, ჩვენ ვირჩევთ $A$ ფორმულის პირველ ძლიერ კვანტორს $(Qx)$, და ზოგადობის შეუზღუდავად ვვარაუდობთ, რომ ის არის სუსტი კვანტორების $(Q_1x_1), \dots, (Q_mx_m), m \geq 0$ საზღვრებში. თუ ჩვენ გავიტანთ წინ ამ კვანტორებს გადატანის წესების გამოყენებით, მაშინ ჩვენ მივიღებთ $A$-ს ექვივალენტურ ფორმულას, $(\exists x_1) \dots (\exists x_m)(\forall x)B$, სადაც $B = A_{-(Qx)(Q_1x_1)\dots(Q_mx_m)}$.[3]

განსაზღვრება 3.2.7-ის თანახმად, $sk((\exists x_1) \dots (\exists x_m)(\forall x)B) = sk((\exists x_1) \dots (\exists x_m)B\sigma)$, სადაც $\sigma$ არის ჩანაცვლება განსაზღვრება 3.2.7-ის მიხედვით (დამტკიცებისათვის არ არის არსებითი $x, m, x_1, \dots, x_m$ ცვლადების შემთხვევების განხილვა).

ინდუქციის ჰიპოთეზის თანახმად, თუ $(\exists x_1) \dots (\exists x_m)B\sigma$ არის ზოგადმართებული, მაშინ $sk((\exists x_1) \dots (\exists x_m)B\sigma)$ აგრეთვე ზოგადმართებულია. ამდენად, საკმარისია ვაჩვენოთ რომ $(\exists x_1) \dots (\exists x_m)(\forall x)B \to (\exists x_1) \dots (\exists x_m)B\sigma$, რომელიც ტრივიალურად ზოგადმართებულია. □

**თეორემა 3.2.4** (სისრულე). *თუ $sk(A)$ არის ზოგადმართებული, მაშინ $A$ აგრეთვე ზოგადმართებულია.*

*დამტკიცება.* ინდუქციით ძლიერი კვანტორების რაოდენობაზე $A$ ფორმულაში. საბაზისო შემთხვევა ტრივიალურია: $sk(A) = A$.

დავუშვათ, რომ თეორემა მოქმედებს ყველა ფორმულაზე, რომელიც შეიცავს არაუმეტეს $n$ ძლიერ კვანტორს და $A$ შეიცავს $n+1$ ძლიერ კვანტორს. მაშინ, ჩვენ ვირჩევთ $A$ ფორმულის პირველ ძლიერ კვანტორს $(Qx)$, და გამოვარჩევთ შემდეგ შემთხვევებს:

- $(Qx)$ არ არის სუსტი კვნატორის საზღვრებში. მაშინ განსაზღვრება 3.2.7-ის თანახმად $sk(A) = sk(A_{-(Qx)}\sigma)$, სადაც $\sigma$ არის $[x \mapsto c]$ (თუ $x \in \mathcal{V}_\mathsf{I}$) ან $[x \mapsto \ulcorner t_1, \dots, t_k \urcorner]$ (თუ $x \in \mathcal{V}_\mathsf{S}$), $c, t_1, \dots, t_k$-სთვის რომლებიც არ გვხვდებიან $A$-ში. ინდუქციის ჰიპოთეზის თანახმად,

[3] შევნიშნოთ, რომ სუსტი კვანტორები ხდებიან არსებობის და ძლიერი კვანტორები უნივერსალური, როდესაც მათ გადავიტანთ ფორმულის წინ.

თუ $sk(A_{-(Qx)}\sigma)$ არის ზოგადმართებული, მაშინ $A_{-(Qx)}\sigma$ არის აგრეთვე ზოგადმართებული. ამდენად, საკმარისია ვაჩვენოთ, რომ თუ $A_{-(Qx)}\sigma$ არის ზოგადმართებული, მაშინ $A$ ასევე ზოგადმართებულია.

შევნიშნოთ, რომ $A$ არის ექვივალენტური $A'$: $(\forall x)A_{-Q(x)}$ (კვანტორების გადატანის წესის გამოყენებით $(Qx)$ გადადის წინ) და ვვარაუდობთ, რომ $A'$ (და აგრეთვე $A$) არ არის ზოგადმართებული. მაშინ, არსებობს ინტერპრეტაცია $\Gamma$, რომელიც არის $A'$-ის კონტრ მოდელი. ინტერპრეტაციის სტანდარტული განსაზღვრების თანახმად, არსებობს ინტერპრეტაცია $\Gamma'$ რომელიც განსხვავდება $\Gamma$-სგან, მხოლოდ $x$-ის შეფასებით და აგრეთვე არის $A'$-ს კონტრ-მოდელი. რადგანაც, $c$ (ან $t_1, \dots, t_k$) არ გვხვდება $A$-ში, ჩვენ შეგვიძლია $\Gamma'$ განვავრცოთ $\Gamma''$-მდე, სადაც $x$ ინტერპრეტირებულია როგორც $c$ (ან $\ulcorner t_1, \dots, t_k \urcorner$). მაშინ $\Gamma''$ იქნება $A_{-(Qx)}\sigma$-ს კონტრ-მოდელი, ეს ნიშნავს რომ$A_{-(Qx)}\sigma$ არ არის ზოგადმართებული.

- $(Qx)$ არის სუსტი კვანტორების $(Q_1x_1), \dots, (Q_mx_m), m > 0$ მოქმედების არეში. ჩვენ გადაგვაქვს ეს კვანტორები ფორმულის წინ კვანტორების გადატანის წესის გამოყენებით, შედეგად ვიღებთ $A$-ს ექვივალენტურ ფორმულას $A'$: $(\exists x_1) \dots (\exists x_m)(\forall x)B$, სადაც $B = A_{-(Qx)(Q_1x_1)\dots(Q_mx_m)}$. განსაზღვრება 3.2.7-ის თანახმად $sk(A') = sk((\exists x_1) \dots (\exists x_m)B\sigma)$, სადაც $\sigma$ არის $[x \mapsto f(x_1, \dots, x_n)]$ (თუ $x \in \mathcal{V}_\mathsf{I}$) და $[x \mapsto \ulcorner f_u^1(x_1, \dots, x_n), \dots, f_u^m(x_1, \dots, x_n) \urcorner]$ (თუ $x \in \mathcal{V}_\mathsf{S}$).

  ინდუქციის ჰიპოთეზის თანახმად, თუ $sk((\exists x_1) \dots (\exists x_m)B\sigma)$ არის ზოგადმართებული, მაშინ $(\exists x_1) \dots (\exists x_m)B\sigma$ არის აგრეთვე ზოგადმართებული. ამრიგად, საკმარისია ვაჩვენოთ, რომ თუ $(\exists x_1) \dots (\exists x_m)B\sigma$ არის ზოგადმართებული, მაშინ $A$ არის აგრეთვე ზოგადმართებული.

  ისევ დავუშვათ, რომ $A'$ არ არის ზოგადმართებული, მაშინ არსებობს ინტერპრეტაცია $\Gamma$ რიმელიც არის $A'$-ის კონტრ-მოდელი და არის $\neg A'$-ს მოდელი. ამორჩევის აქსიომის თანახმად, არსებობს $x_1, \dots, x_m$ ცვლადების ფუნქცია $g$ (ან ფუნქციათა მიმდევრობა $\ulcorner g_u^1, \dots, g_u^n \urcorner$),

რომელიც შეგვიძლია მივანიჭოთ $x$ ცვლადს, და რადგანაც $f$ (ან $\ulcorner f_u^1, \dots, f_u^m \urcorner$) არ გვხვდება $A$-ში, ჩვენ შეგვიძლია ავიღოთ $g = f$ (ან $\ulcorner g_u^1, \dots, g_u^n \urcorner = \ulcorner f_u^1, \dots, f_u^m \urcorner$) და ავაგოთ ინტერპრეტაცია $\Gamma'$, რომელიც იქნება $\Gamma$-ს გაფართოება და $(\forall x_1) \dots (\forall x_m) \neg B\sigma$ ფორმულის მოდელი, ე.ი. $(\exists x_1) \dots (\exists x_m) B\sigma$ ფორმულის კონტრ-მოდელი. ეს ამტკიცებს თეორემას.

□

**შედეგი 3.2.5.** *$sk(A)$ არის შესრულებადად ექვივალუნტური $A$ ფორმულის, სხვაგვარად, $sk(A)$ შესრულებადია მაშინ და მხოლოდ მაშინ, როდესაც $A$ არის შესრულებადი.*

*დამტკიცება.* თეორემების 3.2.3 და 3.2.4 პირდაპირი შედეგი. □

# თავი 4

# დასკვნა და სამომავლო ამოცანები

ავტომატური მსჯელობა ლოგიკისა და კომპიუტერულ მეცნიერებათა ერთ-ერთი ყველაზე მნიშვნელოვანი დარგია. ის ასევე განიხილება როგორც ხელოვნური ინტელექტის ქვედარგი. ეს მიმართულება სწავლობს მსჯელობის სხვადასხვა ასპექტებს. ავტომატური მსჯელობის ყველაზე მნიშვნელოვანი ატრიბუტებია კლასიკური პრედიკატთა აღრიცხვა და ლოგიკის სხვადასხვა კალკულუსები.

ავტომატური მსჯელობის ორი ძირითადი ქვედარგია თეორემათა ავტომატური და ინტერაქციული მტკიცებები. წარმოდგენილი ნაშრომის კვლევის ობიექტიც სწორედ ეს მიმართულებებია. უფრო კონკრეტულად, ჩვენ შევისწავლეთ სქემებისა და ურანგო ლოგიკისათვის თეორემათა დამტკიცების ავტომატური მეთოდები. შემოვიღეთ ურანგო ლოგიკისათვის ტაბლო მეთოდი და შევისწავლეთ ამ ლოგიკაში სკოლემიზაციის პრობლემა.

მიღებული შედეგები როგორც თეორიული, ისე პრაქტიკული ხასიათისაა. ჩვენ ავაგეთ ამოხსნის ახალი პროცედურები სქემებისა და ურანგო ლოგიკისათვის და გავაუმჯობესეთ მათი დამტკიცებათა წარმოდგენის ფორმატები. კვლევის მიმდინარეობისას შექმნილი ალგორითმი და პროცედურა იქნა რეალიზებული და შესაძლებელია მათი პრაქტიკაში გამოყენება.

მიღებული შედეგების თეორიული და პრაქტიკული მნიშვნელობა მეცნიერული დარგების მიმართ ორ ასპექტში უნდა განვიხილოთ. პირველი ასპექტი ჩვენს შემთხვევაში გულისხმობს სქემებისა და ურანგო ლოგიკის მიმართულებების განვითარებასა და გაფართოებას ახალი კალკულუსებისა და მათზე მსჯელობის ალგორითმების შემოღებით.

მეორე ასპექტი მიღებული შედეგების სხვა სფეროებში გამოყენებას უკავშირდება. ამ მხრივ, შედეგები განავრცობენ არსებულს და ქმნიან ახალ ტექნოლოგიებს თეორემათა ავტომატურ მტკიცებაში. მიღებული შედეგები, შესაძლებელია გამოყენებულ იქნას ხელოვნური ინტელექტის ისეთ დარგებში, როგორიცაა *სემანტიკური ქსელი* და *ცოდნის წარმოდგენა*. ერთ-ერთი ასეთი გამოყენების მაგალითი მოყვანილის თავი 3-ში.

როგორც შესავალში იყო აღნიშნული, ლიტერატურაში არსებობს ტაბლო მეთოდის უამრავი გაუმჯობესება. შესაბამისად, ჩვენი სამომავლო მიზანია ურანგო ლოგიკისათვის შემოღებული ტაბლო მეთოდის გაუმჯობესება, მისი გავრცობით ჰიპერ-ტაბლო მეთოდამდე თუ სხვა ტექნოლოგიებით. ეს ამოცანა რა თქმა უნდა არ არის ტრივიალური და საჭიროებს საფუძვლიან კვლევას.

# ლიტერატურა


[ACP09] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. A Schemata Calculus for Propositional Logic. In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 5607 of *Lecture Notes in Computer Science*, pages 32–46, 2009.

[ACP10] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. RegSTAB: A SAT-Solver for Propositional Iterated Schemata. In *International Joint Conference on Automated Reasoning*, pages 309–315, 2010.

[ACP11] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Decidability and Undecidability Results for Propositional Schemata. *Journal of Artificial Intelligence Research*, 40:599–656, 2011.

[AMS98] Serge Autexier, Heiko Mantel, and Werner Stephan. Simultaneous quantifier elimination. pages 141–152, 1998.

[B+97] Bruno Barras et al. *The Coq proof assistant reference manual: Version 6.1*. 1997.

[BCJ+06] Bruno Buchberger, Adrian Craciun, Tudor Jebelean, Laura Kovács, Temur Kutsia, Koji Nakagawa, Florina Piroi, Nikolaj Popov, Judit Robu, Markus Rosenkranz, and Wolfgang Windsteiger. Theorema: Towards computer-aided mathematical theory exploration. *J. Applied Logic*, 4(4):470–504, 2006.

[BEL01] Matthias Baaz, Uwe Egly, and Alexander Leitsch. Normal form transformations. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 273–333. North Holland, 2001.

[BMS10] David Baelde, Dale Miller, and Zachary Snow. Focused inductive theorem proving. pages 278–292, 2010.

[Bol99] Harold Boley. *A Tight, Practical Integration of Relations and Functions*, volume 1712 of *Lecture Notes in Computer Science*. Springer, 1999.

[BP10] Christoph Benzmüller and Adam Pease. Progress in automating higher-order ontology reasoning. In *Proceedings of the Second International Workshop on Practical Aspects of Automated Reasoning*, 2010.

[BTPF08] Christoph Benzmüller, Frank Theiss, Larry Paulson, and Arnaud Fietzke. LEO-II - a cooperative automatic theorem prover for higher-order logic. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008*, volume 5195 of *LNCS*, pages 162–170. Springer, 2008.

[Buc96] Bruno Buchberger. Mathematica as a rewrite language. In Tetsuo Ida, Atsushi Ohori, and Masato Takeich, editors, *Functional and Logic Programming - 2nd Fuji International Workshop, FLOPS 1996, Shonan Village, Japan, June 4-6, 1996. Proceedings*, Lecture Notes in Computer Science, pages 1–13. Springer, 1996.

[Bus95] S. R. Buss. On herbrand's theorem. In *Logic and Computational Complexity*, volume 960 of *LNCS*, pages 195–209. Springer Berlin Heidelberg, 1995.

[Bus98] S. R. Buss. *Handbook of Proof Theory*. Elsevier, 1998.

[CD97] Eric Chasseur and Yves Deville. Logic program schemas, constraints, and semi-unification. In Norbert E. Fuchs, editor, *Logic Programming Synthesis and Transformation, 7th International Workshop, LOPSTR'97, Leuven, Belgium, July 10-12, 1997, Proceedings*, volume 1463 of *Lecture Notes in Computer Science*, pages 69–89. Springer, 1997.

[CF04] Jorge Coelho and Mário Florido. CLP(Flex): Constraint Logic Programming applied to XML processing. In Robert Meersman and Zahir

Tari, editors, *CoopIS/DOA/ODBASE (2)*, volume 3291 of *LNCS*, pages 1098–1112. Springer, 2004.

[CF06] Jorge Coelho and Mário Florido. Veriflog: A constraint logic programming approach to verification of website content. In Heng Tao Shen, Jinbao Li, Minglu Li, Jun Ni, and Wei Wang, editors, *Advanced Web and Network Technologies, and Applications, APWeb 2006 International Workshops: XRA, IWSN, MEGA, and ICSE, Harbin, China, January 16-18, 2006, Proceedings*, volume 3842 of *Lecture Notes in Computer Science*, pages 148–156. Springer, 2006.

[CF07] Jorge Coelho and Mário Florido. XCentric: logic programming for XML processing. In Irini Fundulaki and Neoklis Polyzotis, editors, *9th ACM International Workshop on Web Information and Data Management (WIDM 2007), Lisbon, Portugal, November 9, 2007*, pages 1–8. ACM, 2007.

[CFK07] Jorge Coelho, Mário Florido, and Temur Kutsia. Sequence disunification and its application in collaborative schema construction. In Mathias Weske, Mohand-Said Hacid, and Claude Godart, editors, *WISE Workshops*, volume 4832 of *Lecture Notes in Computer Science*, pages 91–102. Springer, 2007.

[CFK09] Jorge Coelho, Mário Florido, and Temur Kutsia. Collaborative schema construction using regular sequence types. In *Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2009, 10-12 August 2009, Las Vegas, Nevada, USA*, pages 290–295. IEEE Systems, Man, and Cybernetics Society, 2009.

[CHK91] Hong Chen, Jieh Hsiang, and Hwa-Chung Kong. On finite representations of infinite sequences of terms. In S. Kaplan and M. Okada, editors, *Conditional and Typed Rewriting Systems*, volume 516 of *Lecture Notes in Computer Science*, pages 99–114. Springer Berlin Heidelberg, 1991.

[CKR16a] Gela Chankvetadze, Lia Kurtanidze, and Mikheil Rukhaia. Proof Construction in Unranked Logic. In Temur Jangveladze and Zurab Kiguradze, editors, *Reports of Enlarged Sessions of the Seminar of I. Vekua Institute of Applied Mathematics*, volume 30, pages 11–14. 2016.

[CKR16b] Gela Chankvetadze, Lia Kurtanidze, and Mikheil Rukhaia. Semi-Automated Construction of Proof Schemata. In *Journal of Applied Mathematics, Informatics and Mechanics*, volume 21, pages 83–91. 2016.

[CLs07] Information technology – Common Logic (CL): a framework for a family of logic-based languages, 2007. International Standard ISO/IEC 24707, 1st edn.

[DGHP13] Marcello D'Agostino, Dov M Gabbay, Reiner Hähnle, and Joachim Posegga. *Handbook of tableau methods*. Springer Science & Business Media, 2013.

[DKR17] Besik Dundua, Lia Kurtanidze, and Mikheil Rukhaia. Unranked Tableaux Calculus and its Web-related Applications. In *IEEE Conference on Electrical and Computer Engineering*, pages 1181–1184. IEEE Xplore Digital Library, 2017.

[DLL+12] Cvetan Dunchev, Alexander Leitsch, Tomer Libal, Martin Riener, Mikheil Rukhaia, Daniel Weller, and Bruno Woltzenlogel-Paleo. System Feature Description: Importing Refutations into the GAPT Framework. In David Pichardie and Tjark Weber, editors, *Second International Workshop on Proof Exchange for Theorem Proving (PxTP 2012)*, volume 878 of *CEUR Workshop Proceedings*, pages 51–57, 2012.

[DLL+13] Cvetan Dunchev, Alexander Leitsch, Tomer Libal, Martin Riener, Mikheil Rukhaia, Daniel Weller, and Bruno Woltzenlogel-Paleo.

ProofTool: a GUI for the GAPT Framework. In Cezary Kaliszyk and Christoph Lüth, editors, *Proceedings 10th International Workshop On User Interfaces for Theorem Provers (UITP 2012)*, volume 118 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–14, 2013.

[DLRW12] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. CERES for Propositional Proof Schemata. Technical report, Vienna University of Technology, 2012.

[DLRW15] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. Cut-elimination and proof schemata. In Martin Aher, Daniel Hole, Emil Jeřábek, and Clemens Kupke, editors, *Logic, Language, and Computation*, volume 8984 of *Lecture Notes in Computer Science*, pages 117–136. Springer Berlin Heidelberg, 2015.

[EHR+16] Gabriel Ebner, Stefan Hetzl, Giselle Reis, Martin Riener, Simon Wolfsteiner, and Sebastian Zivota. System description: Gapt 2.0. In *Automated Reasoning: 8th International Joint Conference, IJCAR 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*, pages 293–301. Springer International Publishing, 2016.

[Gen35a] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39:176–210, 1935.

[Gen35b] Gerhard Gentzen. Untersuchungen über das logische Schließen II. *Mathematische Zeitschrift*, 39:405–431, 1935.

[Gen98] Michael R. Genesereth. Knowledge Interchange Format, draft proposed American National Standard (dpANS). Technical Report NCITS.T2/98-004, 1998. Available from http://logic.stanford.edu/kif/dpans.html.

[Gin91] Matthew L. Ginsberg. The MVL theorem proving system. *SIGART Bull.*, 2(3):57–60, 1991.

[Gir91] Jean-Yves Girard. A new constructive logic: classical logic. 1:255–296, 1991.

[HA28] David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik*. J. Springer, 1928.

[Ham97] Makoto Hamana. Term rewriting with sequences. In *Proceedings of the First International Theorema Workshop*, number 97-20 in RISC Technical Report series, Hagenberg, Austria, 1997.

[HLR+14] Stefan Hetzl, Alexander Leitsch, Giselle Reis, Janos Tapolczai, and Daniel Weller. Introducing quantified cuts in logic with equality. In *Automated Reasoning*, volume 8562 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2014.

[HLRR13] Stefan Hetzl, Tomer Libal, Martin Riener, and Mikheil Rukhaia. Understanding Resolution Proofs through Herbrand's Theorem. In Didier Galmiche and Dominique Larchey-Wendling, editors, *Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux 2013)*, volume 8123 of *Lecture Notes in Computer Science*, pages 157–171, 2013.

[HM01a] Volker Haarslev and Ralf Möller. Racer system description. In *International Joint Conference on Automated Reasoning*, pages 701–705. Springer, 2001.

[HM01b] Pat J. Hayes and Christopher Menzel. Semantics of Knowledge Interchange Format. http://reliant.teknowledge.com/IJCAI01/HayesMenzel-SKIF-IJCAI2001.pdf, 2001.

[HM05] Pat J. Hayes and Christopher Menzel. Simple Common Logic. In *W3C Workshop on Rule Languages for Interoperability*. W3C, 2005.

[HV06] Ian Horrocks and Andrei Voronkov. Reasoning support for expressive ontology languages using a theorem prover. In *Foundations of Information and Knowledge Systems*, pages 201–218. Springer, 2006.

[JR08] Florent Jacquemard and Michaël Rusinowitch. Closure of hedge-automata languages by hedge rewriting. In Andrei Voronkov, editor, *RTA*, volume 5117 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2008.

[KB04] Temur Kutsia and Bruno Buchberger. Predicate logic with sequence variables and sequence function symbols. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *MKM*, volume 3119 of *LNCS*, pages 205–219. Springer, 2004.

[KM05] Temur Kutsia and Mircea Marin. Can context sequence matching be used for querying XML? In Laurent Vigneron, editor, *Proceedings of the 19th International Workshop on Unification UNIF'05*, pages 77–92, Nara, Japan, 22 April 2005.

[KM12] Temur Kutsia and Mircea Marin. Solving, reasoning, and programming in common logic. In *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2012, Timisoara, Romania, September 26-29, 2012*, pages 119–126. IEEE Computer Society, 2012.

[KR15] Lia Kurtanidze and Mikheil Rukhaia. About Correspondence Between Proof Schemata and Unranked Logics. In Temur Jangveladze and Zurab Kiguradze, editors, *Reports of Enlarged Sessions of the Seminar of I. Vekua Institute of Applied Mathematics*, volume 29, pages 72–75. 2015.

[Kut03] Temur Kutsia. Equational prover of THEOREMA. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications, 14th International Conference, RTA 2003, Valencia, Spain, June 9-11, 2003, Proceed-*

*ings*, volume 2706 of *Lecture Notes in Computer Science*, pages 367–379. Springer, 2003.

[Kut07] Temur Kutsia. Solving equations with sequence variables and sequence functions. *J. Symb. Comput.*, 42(3):352–388, 2007.

[Lei97] Alexander Leitsch. *The resolution calculus*. Texts in theoretical computer science. Springer-Verlag Inc., New York, NY, USA, 1997.

[LM09] Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46):4747–4768, 2009.

[LRR14] Tomer Libal, Martin Riener, and Mikheil Rukhaia. Advanced Proof Viewing in ProofTool. In Christoph Benzmüller and Bruno Woltzenlogel Paleo, editors, *Proceedings Eleventh Workshop on User Interfaces for Theorem Provers, (UITP 2014)*, volume 167 of *Electronic Proceedings in Theoretical Computer Science*, pages 35–47, 2014.

[McC10] W. McCune. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.

[Men11] Christopher Menzel. Knowledge representation, the World Wide Web, and the evolution of logic. *Synthese*, 182(2):269–295, 2011.

[MK03] Mircea Marin and Temur Kutsia. On the implementation of a rule-based programming system and some of its applications. In Boris Konev and Renate Schmidt, editors, *Proceedings of the 4th International Workshop on the Implementation of Logics*, pages 55–69, Almaty, Kazakhstan, 2003.

[MK06] Mircea Marin and Temur Kutsia. Foundations of the rule-based system $\rho$Log. *Journal of Applied Non-Classical Logics*, 16(1-2):151–168, 2006.

[MT03] Mircea Marin and Dorin Tepeneu. Programming with sequence variables: The Sequentica package. In Peter Mitic, Philip Ramsden, and Janet Carne, editors, *Challenging the Boundaries of Symbolic Computation. Proceedings of 5th International Mathematica Symposium*, pages 17–24, London, 2003. Imperial College Press.

[NPW02] Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer, 2002.

[OSV10] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala: A Comprehensive Step-by-step Guide*. Artima, Inc., 2nd edition, 2010.

[Pau90] Lawrence C. Paulson. Isabelle: The next 700 theorem provers. In *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[Pfe99] Frank Pfenning. Automated theorem proving. *Course Notes on the WWW*, 1999.

[RF97] Julian Richardson and Norbert E. Fuchs. Development of correct transformation schemata for Prolog programs. In Norbert E. Fuchs, editor, *Logic Programming Synthesis and Transformation, 7th International Workshop, LOPSTR'97, Leuven, Belgium, July 10-12, 1997, Proceedings*, volume 1463 of *Lecture Notes in Computer Science*, pages 263–281. Springer, 1997.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965.

[Ruk13] Mikheil Rukhaia. *About Cut-Elimination in Schematic Proofs. A monograph*. Lambert Academic Publishing, Saarbrücken, 2013.

[RV01a] Alan JA Robinson and Andrei Voronkov. *Handbook of automated reasoning*, volume 1. Elsevier, North Holland, 2001.

[RV01b] Alan JA Robinson and Andrei Voronkov. *Handbook of automated reasoning*, volume 2. Elsevier, North Holland, 2001.

[RV02] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15(2,3):91–110, 2002.

[Sch04a] Sebastian Schaffert. *Xcerpt: a rule-based query and transformation language for the web*. PhD thesis, University of Munich, 2004.

[Sch04b] S. Schulz. System Description: E 0.81. In D. Basin and M. Rusinowitch, editors, *Proc. of the 2nd IJCAR, Cork, Ireland*, volume 3097 of *LNAI*, pages 223–228. Springer, 2004.

[Smu68] Raymond Smullyan. *First-Order Logic*. Springer, 1968.

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[Sut09] Geoff Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[Tak87] Gaisi Takeuti. *Proof Theory*. North Holland, second edition, 1987.

[TH06] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In *International Joint Conference on Automated Reasoning*, pages 292–297. Springer, 2006.

[WB01] Manfred Widera and Christoph Beierle. A term rewriting scheme for function symbols with variable arity. Technical Report 280, Praktische Informatik VIII, FernUniversität Hagen, Germany, 2001.

[WDF+09] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, and Wischnewski. P. SPASS version 3.5. *in 22nd International Conference on Automated Deduction, CADE 2009, LNCS 5663, pp. 140-145.*, 2009.

[Wol03] Stephen Wolfram. *The Mathematica Book*. Wolfram Media, 5th edition, 2003.

[WR12] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*, volume II. 1912.

# ინდექსი