Proof Verification and Proof Discovery for Relativity[*]

Naveen Sundar G. • Selmer Bringsjord • Joshua Taylor
govinn@rpi • selmer@rpi.edu • tayloj@rpi.edu
Department of Computer Science
Department of Cognitive Science
The Rensselaer AI & Reasoning (RAIR) Lab
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA

June 11, 2012

Contents

1 The Vision **1**

2 Background **1**

3 Systems for Proof Verification and Proof Discovery **2**
3.1 Proof Verification using Denotational Proof Languages 2
3.2 Semi-automated Proof Discovery Using Automated Theorem Proving 2
3.3 Proof Verification and Proof Discovery in Slate 3

4 Examples **3**
4.1 Theorem Neat: No Event at Two Places 3

5 Going Forward **4**
5.1 Reasoning with the Field Axioms 4
5.2 Visual Reasoning 5

[*] We are deeply grateful for support received from both the John Templeton Foundation and the Air Force Office of Scientific Research.

1 The Vision

The vision of machines carrying out substantive proof verification, proof discovery, and theorem discovery (= conjecture generation) in mathematics and the natural sciences has a long history that reaches back before the development of automatic systems for such purposes.[1] While there has been considerable progress in proof verification in mathematics, e.g. the Mizar project (14) (and see also (7) for coverage of the Four-Color Theorem, now machine verified), there has been little such work carried out in the realm of the natural sciences — until recently. This delay in the case of the natural sciences can be attributed to both a lack of formal analysis of the "theories" in such sciences, and the lack of sufficient progress in automated theorem proving. While the lack of analysis is due to an inclination toward informality and empiricism on the part of nearly all of the relevant scientists, the lack of progress is to be expected, given the computational hardness of automated theorem proving; after all, theoremhood in even first-order logic is Turing-undecidable.

To be sure, there has been some work devoted to formally modeling biological theories; e.g., Woodger and Tarski, as early as 1937, carried out such work (18). In addition, a representative of the modern logicist analysis of the structure of biological entities, a precursor to the formal, computational theory development we are pursuing, is the Open Biomedical Ontologies Foundry enterprise described in (15). In the case of physics, the situation is much more mature, courtesy of extensive, seminal investigation by Székely, Andréka, X. Madarász and Németi in (13; 1; 2; 17; 3), which is devoted to setting out the logical structure of relativistic theories in physics. This is a necessary first step toward the trio of proof verification, proof discovery, and theorem discovery in physics; toward, that is, a point in time when physics and AI are inseparable and synergistic. We give in the present short paper a compressed report on our attempt to build upon these formal theories using logic-based AI in order to achieve, in relativity, both machine proof verification and machine proof discovery.

2 Background

Given an informal theory T, we wish to formalize it as a logical theory Γ in a formal language $\mathcal{L}$. In our case, the theory is special relativity, *SpecRel*, and the language is $\mathcal{L}$, that of first-order logic. We also assume that there is a proof theory $\vdash$ associated with the language (with suitable augmentation of alphabet and grammar to enable mechanically checkable inferences). Truth of any formula in $\mathcal{L}$ is defined through standard model-theoretic interpretations over first-order structures. That a sentence ϕ is true under an interpretation M is written $M \models \phi$. Provability of ϕ from a set of axioms Γ is written $\Gamma \vdash_\rho \phi$, where ρ is one possible proof of ϕ from Γ. If we are not interested in any one proof, but rather want to look at provability in general, we write $\Gamma \vdash \phi$. We use terminology closely following that used in Boolos et al. in (10).

With this background in place, we now define the following terms:

Proof Verification Given a proof ρ, a formula ϕ, and a set of axioms Γ, verify that ρ represents a proof of ϕ from Γ. Expressed in query form: Is it true that $\Gamma \vdash_\rho \phi$?

Proof Discovery Given a formula ϕ and a set of axioms Γ, find a proof ρ that derives ϕ from Γ. I.e., find a ρ such that $\Gamma \vdash_\rho \phi$.

Theorem Discovery Given Γ, find an *interesting* ϕ such that $\Gamma \vdash \phi$. (This is also known as *conjecture generation*.)

[1] One could in fact argue that Aristotle, over two thousand years back, intuitively had, and indeed pursued, the vision. See the discussion of Aristotle in connection with modern knowledge-based AI given by Glymour in (12).

Two things should be noted. First, there is no mechanism to represent ρ in $\mathcal{L}$ in a transparent fashion. This is why denotational proof languages are very useful. Second, and this will come as no surprise to cognoscenti, the problem of what makes a theorem interesting is ill-defined. We do not discuss interestingness any further in the present paper, but leave the topic to subsequent expansion of the work described synoptically herein.

3 Systems for Proof Verification and Proof Discovery

In the section, we briefly enumerate the formalisms and systems that we will be using. The proofs shown in this paper were done using the Slate system (11).

3.1 Proof Verification using Denotational Proof Languages

For proof verification, we use a system built upon a denotational proof language (DPL). Given a logical system $\mathcal{L}$, a DPL $\mathcal{K}$ for that system provides a way to formally express proofs in $\mathcal{L}$ in the language of $\mathcal{K}$. The syntax of a DPL is built using the $\lambda\mu$ calculus; the basic syntactic categories are *propositions*, *deductions*, and *methods*. Evaluation in a DPL reduces to proof checking. A Deduction D, when evaluated correctly in the context of an assumption base β, which is a set of propositions corresponding to axioms or premises, produces a proposition P, denoted as $\beta \vdash D \rightsquigarrow P$. If the evaluation is not warranted, the object error, which differs from all the propositions, is produced. Deductions are either a single method or methods combined in permissible ways specified by the proof system of $\mathcal{L}$.

Methods can be *primitive* or *derived*. Primitive methods correspond to primitive rules of reasoning in $\mathcal{L}$. Derived methods in the $\lambda\mu$ calculus, or $\lambda\mu$-methods, are arbitrary abstractions over deductions and are analogous to functions in the λ calculus. $\lambda\mu$-methods are intended to abstract over common reasoning scenarios and allow for modular theorem proving.[2] Derived methods are obtained from primitive methods through a certain set of operations; e.g., method composition. We use the DPL *CND* described in (4, Chapter 4); this DPL is a formalization of classical natural deduction.

3.2 Semi-automated Proof Discovery Using Automated Theorem Proving

Inspired by how a user of a monolithic theorem prover can break up problems, we have built a simple inference system **semi** that models semi-automated theorem proving with a sound and complete monolithic theorem prover.

Given a first-order language $\mathcal{L}$ whose sentences are $S(\mathcal{L})$, let $\beta \subset S(\mathcal{L})$ be the entire set of axioms that one is allowed to use. In our case, β will be the field axioms, axioms in physics, and any other definitional axioms. We define a *problem* to be a triple consisting of a *theorem* to be proved, a list of *premises*, and a list of other problems whose theorems can be added to the list of premises of this problem. Formally,

$$\begin{aligned}
&\mathsf{problem} := \langle \mathsf{theorem}, \mathsf{premises}, \texttt{list}\langle \mathsf{problem}\rangle\rangle\\
&\mathsf{theorem} \in S(\mathcal{L})\\
&\mathsf{premises} \subset S(\mathcal{L})
\end{aligned}$$

We have the function *theorem*, which denotes the theorem to be proved in a problem.

$$\texttt{eval}(\langle \mathsf{theorem}, \mathsf{premises}, \texttt{nil}\rangle) = \begin{cases} \mathit{error} \text{ if premises} \not\subseteq \beta \\ \mathit{prover}(\mathit{theorem}, \mathit{premises}) \text{ if premises} \subseteq \beta \end{cases}$$

$$\texttt{eval}(\langle \mathsf{theorem}, \mathsf{premises}, [p_1, \ldots, p_n]\rangle) = (\wedge_i \texttt{eval}(p_i)) \Rightarrow \mathit{prover}(\mathit{theorem}, \mathit{premises} \cup \{\mathit{theorem}(p_1), \ldots, \mathit{theorem}(p_n)\}))$$

[2] For more on DPLs see Arkoudas's (4). For an example of methods used to produce cognitively "deep" proofs, see (6).

Note that one can directly embed a theorem prover in a DPL if a primitive method denoting the theorem prover is introduced. This can allow us to combine different proof systems, say a resolution-based theorem prover with a natural-deduction DPL; the combination yields a hybrid DPL. The **semi** system is more limited, in that the only rules of inferences introduced are those of the theorem prover's. Such an oracular black-box system is useful in helping us decide quickly, for instance, if a certain lemma or sub proof can be fruitful in a bigger proof.

3.3 Proof Verification and Proof Discovery in Slate

Slate is a graphical workspace for constructing natural deduction proofs in propositional logic, first-order logic and modal logics. Slate also has the ability to automatically discover proofs by calling the Snark theorem prover (see (16).) For an overview of an earlier version of Slate, please see (16). Formally, the current version of Slate can be seen as a graphical DPL with primitive methods for natural deduction and a primitive method for reasoning by Snark.

4 Examples

We report upon nascent work in producing machine-verifiable proofs and machine-generated proofs for theorems in (2; 3). Our initial work focuses on the special theory of relativity, which has been formalized as the set of axioms $\mathsf{SpecRel} = \{\mathsf{AxFd}, \mathsf{AxPh}, \mathsf{AxEv}, \mathsf{AxSf}, \mathsf{AxSm}\}$.

4.1 Theorem Neat: No Event at Two Places

We have obtained a machine-verified proof and a machine-generated proof of a simple theorem in Slate whose givens include lemmas from field theory. The theorem, dubbed 'Neat,' states that for no inertial observer are the events at one point the same as the events at another point. This theorem depends only the field axioms and the physics axiom AxPh; the latter states that the speed of light for a given inertial observer is the same in every direction and at every point. Our formalization of this axiom follows closely follows that in (3), and the language is the same, save for:

1. Qq: a predicate symbol representing objects in $\mathsf{Q} \times \mathsf{Q} \times \mathsf{Q} \times \mathsf{Q}$;
2. In: a predicate symbol defined below for representing events;
3. speed: a field-theoretic functor for representing the speed between two space-time coordinates;
4. c: the functor representing the light speed for a given observer.

The axiom AxPh is given below:

$$\forall m\ \forall xy(\mathsf{IOb}(m) \land \mathsf{Qq}(x)\mathsf{Qq}(y)) \Rightarrow \big((\exists p\ \mathsf{Ph}(p) \land \mathsf{W}(m,p,x) \land \mathsf{W}(m,p,y)) \Leftrightarrow \mathsf{speed}(x,y) = \mathsf{c}(m)\big)$$

We introduce a new functor $\mathsf{ev}(m,x)$ for capturing the set of events observed by an observer m at a space-time point x. This is defined using the W predicate symbol and a new predicate symbol In:

$$\forall m,b,p\ (\mathsf{IOb}(m) \land \mathsf{B}(b) \land \mathsf{Qq}(p)) \Rightarrow \mathsf{In}(b, \mathsf{ev}(m,p)) \Leftrightarrow \mathsf{W}(m,b,p)$$

Theorem Neat is then:

$$\forall m\ \forall xy(\mathsf{IOb}(m) \land \mathsf{Qq}(x) \land \mathsf{Qq}(y)) \Rightarrow \big(x \neq y \Rightarrow \mathsf{ev}(m,x) \neq \mathsf{ev}(m,y)\big)$$

Figure 1 shows a semi-automated proof of Theorem Neat in Slate. Since the version of Slate we employ is single-sorted, we use sort predicates and axioms delineating the sorts. We also use a lemma from field theory: Speed2. The lemma states that given two distinct points, there is always another point which can be reached at the speed of light from one of the points but not the other.

$$\forall m \forall xy(\mathsf{IOb}(m) \land \mathsf{Qq}(x) \land \mathsf{Qq}(y)) \Rightarrow \big(x \neq y \Rightarrow \exists z(\mathsf{Q}(z) \land \mathsf{speed}(z,x) = c(m) \land \mathsf{speed}(z,y) \neq c(m))\big)$$

Figure 2 shows the resolution and paramodulation proofs from Snark of the lemmas and the theorem. The images are rotated to aid in reading. Figure 3 shows the full manual proof using natural deduction in Slate.[3] The manual proof shown here uses only some of the sorts.

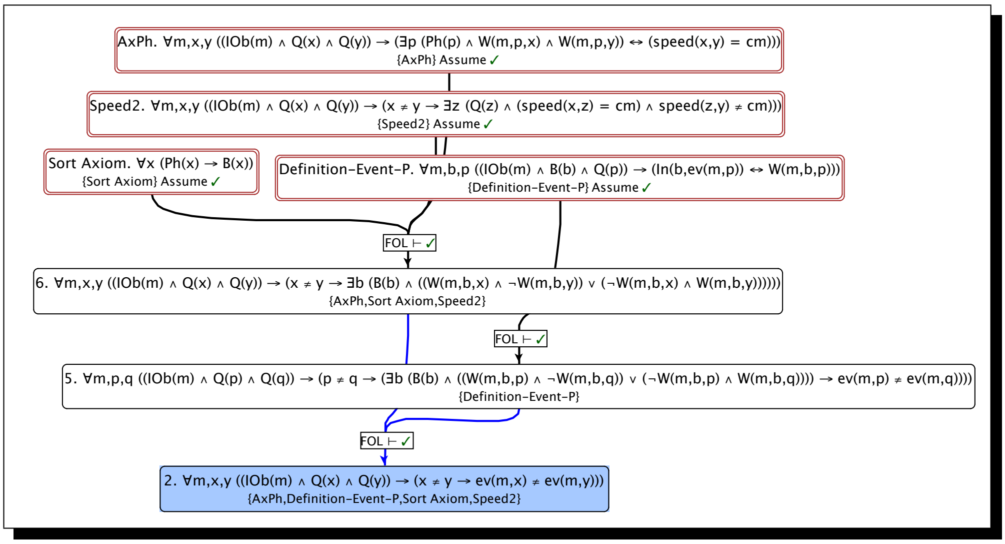

AxPh. ∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (∃p (Ph(p) ∧ W(m,p,x) ∧ W(m,p,y)) ↔ (speed(x,y) = cm)))
{AxPh} Assume ✓
Speed2. ∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ∃z (Q(z) ∧ (speed(x,z) = cm) ∧ speed(z,y) ≠ cm)))
{Speed2} Assume ✓
Sort Axiom. ∀x (Ph(x) → B(x))
{Sort Axiom} Assume ✓
Definition–Event–P. ∀m,b,p ((IOb(m) ∧ B(b) ∧ Q(p)) → (In(b,ev(m,p)) ↔ W(m,b,p)))
{Definition–Event–P} Assume ✓
FOL ⊢ ✓
6. ∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ∃b (B(b) ∧ ((W(m,b,x) ∧ ¬W(m,b,y)) ∨ (¬W(m,b,x) ∧ W(m,b,y))))))
{AxPh,Sort Axiom,Speed2}
FOL ⊢ ✓
5. ∀m,p,q ((IOb(m) ∧ Q(p) ∧ Q(q)) → (p ≠ q → (∃b (B(b) ∧ ((W(m,b,p) ∧ ¬W(m,b,q)) ∨ (¬W(m,b,p) ∧ W(m,b,q)))) → ev(m,p) ≠ ev(m,q))))
{Definition–Event–P}
FOL ⊢ ✓
2. ∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ev(m,x) ≠ ev(m,y)))
{AxPh,Definition–Event–P,Sort Axiom,Speed2}

Figure 1: Semi-Automated Slate Proof of Neat

5 Going Forward

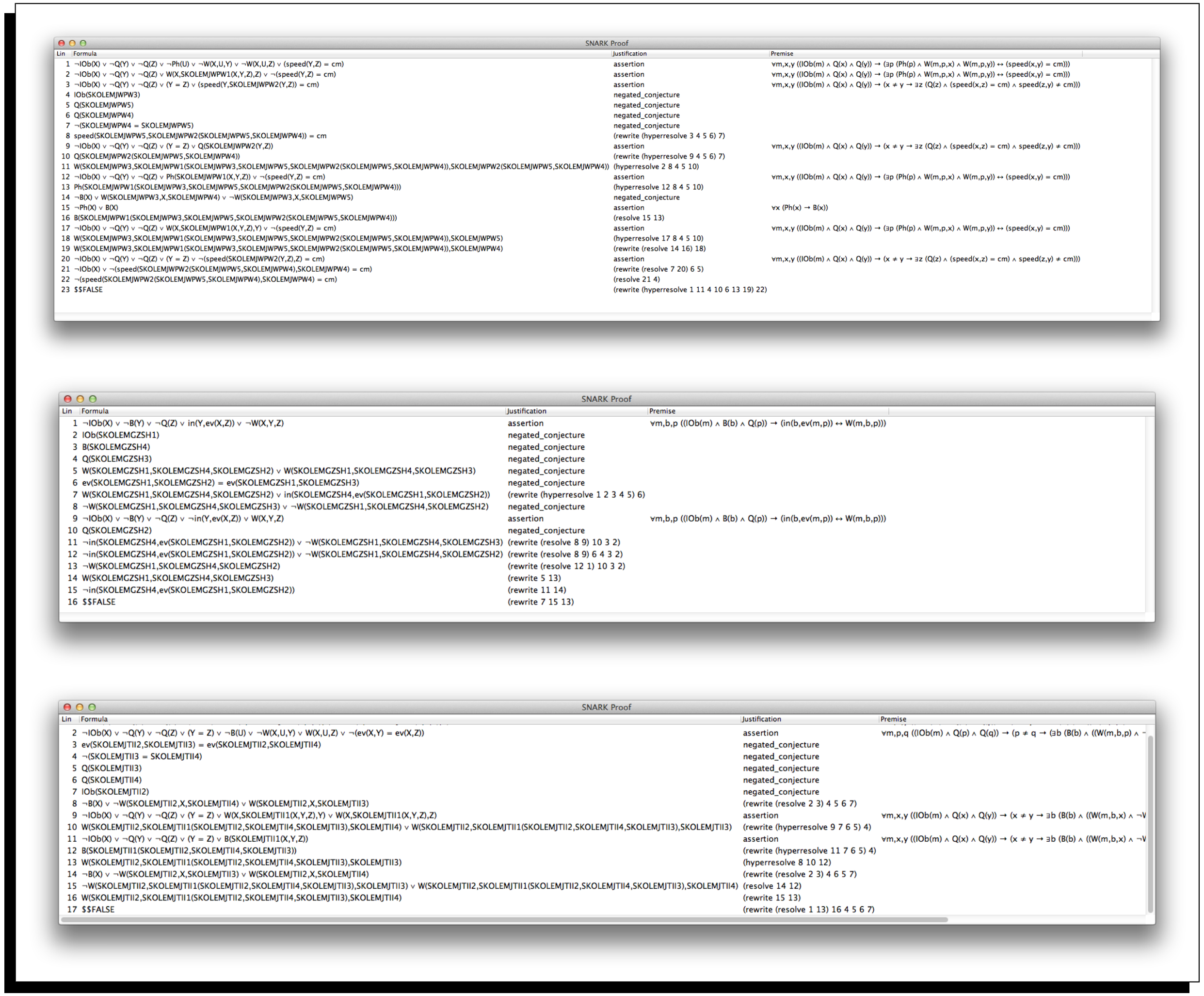
To prove lemmas as complex as Speed2 from field theory, and to tackle more interesting physics theorems, two specific extensions are needed. In the short term, machinery is needed for efficiently reasoning with fields; and in the long term, visual and diagrammatic proofs have to be accommodated.[4] At the conference, we will report upon progress made in these two directions.

5.1 Reasoning with the Field Axioms

Even though the formal language developed in (2; 3) is concise, the proofs given make use of concepts which are not directly defined in the formal language. This includes, for example: lines, slopes of lines, planes, intersections, vectors, etc. Work is in progress to develop a signature σ_{geom} for geometric concepts and a set of DPL methods $F_{geom} = \{f_1, \ldots, f_n\}$ for geometric reasoning directly using the field axioms. Such abstractions are needed if human-readable proofs are to be developed. For instance, the proof of Theorem 2.1 (Theorem NFTLIO) in (3) has a significant amount of geometric content in

[3] A bigger version of the manual proof can be obtained at http://www.cs.rpi.edu/~govinn/files/Theorem_no_event_at_two_place_manual.pdf.

[4] A diagrammatic reasoning system, Vivid, is presented in (8).



SNARK Proof

Lin	Formula	Justification	Premise
1	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ ¬Ph(U) ∨ ¬W(X,U,Y) ∨ ¬W(X,U,Z) ∨ (speed(Y,Z) = cm)	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (∃p (Ph(p) ∧ W(m,p,x) ∧ W(m,p,y)) ↔ (speed(x,y) = cm)))
2	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ W(X,SKOLEMJWPW1(X,Y,Z),Z) ∨ ¬(speed(Y,Z) = cm)	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (∃p (Ph(p) ∧ W(m,p,x) ∧ W(m,p,y)) ↔ (speed(x,y) = cm)))
3	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ (Y = Z) ∨ (speed(Y,SKOLEMJWPW2(Y,Z)) = cm)	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ∃z (Q(z) ∧ (speed(x,z) = cm) ∧ speed(z,y) ≠ cm)))
4	IOb(SKOLEMJWPW3)	negated_conjecture	
5	Q(SKOLEMJWPW5)	negated_conjecture	
6	Q(SKOLEMJWPW4)	negated_conjecture	
7	¬(SKOLEMJWPW4 = SKOLEMJWPW5)	negated_conjecture	
8	speed(SKOLEMJWPW5,SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4)) = cm	(rewrite (hyperresolve 3 4 5 6) 7)	
9	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ (Y = Z) ∨ Q(SKOLEMJWPW2(Y,Z))	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ∃z (Q(z) ∧ (speed(x,z) = cm) ∧ speed(z,y) ≠ cm)))
10	Q(SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4))	(rewrite (hyperresolve 9 4 5 6) 7)	
11	W(SKOLEMJWPW3,SKOLEMJWPW1(SKOLEMJWPW3,SKOLEMJWPW5,SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4)),SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4))	(hyperresolve 2 8 4 5 10)	
12	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ Ph(SKOLEMJWPW1(X,Y,Z)) ∨ ¬(speed(Y,Z) = cm)	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (∃p (Ph(p) ∧ W(m,p,x) ∧ W(m,p,y)) ↔ (speed(x,y) = cm)))
13	Ph(SKOLEMJWPW1(SKOLEMJWPW3,SKOLEMJWPW5,SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4)))	(hyperresolve 12 8 4 5 10)	
14	¬B(X) ∨ W(SKOLEMJWPW3,X,SKOLEMJWPW4) ∨ ¬W(SKOLEMJWPW3,X,SKOLEMJWPW5)	negated_conjecture	
15	¬Ph(X) ∨ B(X)	assertion	∀x (Ph(x) → B(x))
16	B(SKOLEMJWPW1(SKOLEMJWPW3,SKOLEMJWPW5,SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4)))	(resolve 15 13)	
17	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ W(X,SKOLEMJWPW1(X,Y,Z),Y) ∨ ¬(speed(Y,Z) = cm)	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (∃p (Ph(p) ∧ W(m,p,x) ∧ W(m,p,y)) ↔ (speed(x,y) = cm)))
18	W(SKOLEMJWPW3,SKOLEMJWPW1(SKOLEMJWPW3,SKOLEMJWPW5,SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4)),SKOLEMJWPW5)	(hyperresolve 17 8 4 5 10)	
19	W(SKOLEMJWPW3,SKOLEMJWPW1(SKOLEMJWPW3,SKOLEMJWPW5,SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4)),SKOLEMJWPW4)	(rewrite (resolve 14 16) 18)	
20	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ (Y = Z) ∨ ¬(speed(SKOLEMJWPW2(Y,Z),Z) = cm)	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ∃z (Q(z) ∧ (speed(x,z) = cm) ∧ speed(z,y) ≠ cm)))
21	¬IOb(X) ∨ ¬(speed(SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4),SKOLEMJWPW4) = cm)	(rewrite (resolve 7 20) 6 5)	
22	¬(speed(SKOLEMJWPW2(SKOLEMJWPW5,SKOLEMJWPW4),SKOLEMJWPW4) = cm)	(resolve 21 4)	
23	$$FALSE	(rewrite (hyperresolve 1 11 4 10 6 13 19) 22)	

SNARK Proof

Lin	Formula	Justification	Premise
1	¬IOb(X) ∨ ¬B(Y) ∨ ¬Q(Z) ∨ in(Y,ev(X,Z)) ∨ ¬W(X,Y,Z)	assertion	∀m,b,p ((IOb(m) ∧ B(b) ∧ Q(p)) → (in(b,ev(m,p)) ↔ W(m,b,p)))
2	IOb(SKOLEMGZSH1)	negated_conjecture	
3	B(SKOLEMGZSH4)	negated_conjecture	
4	Q(SKOLEMGZSH3)	negated_conjecture	
5	W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH2) ∨ W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH3)	negated_conjecture	
6	ev(SKOLEMGZSH1,SKOLEMGZSH2) = ev(SKOLEMGZSH1,SKOLEMGZSH3)	negated_conjecture	
7	W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH2) ∨ in(SKOLEMGZSH4,ev(SKOLEMGZSH1,SKOLEMGZSH2))	(rewrite (hyperresolve 1 2 3 4 5) 6)	
8	¬W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH3) ∨ ¬W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH2)	negated_conjecture	
9	¬IOb(X) ∨ ¬B(Y) ∨ ¬Q(Z) ∨ ¬in(Y,ev(X,Z)) ∨ W(X,Y,Z)	assertion	∀m,b,p ((IOb(m) ∧ B(b) ∧ Q(p)) → (in(b,ev(m,p)) ↔ W(m,b,p)))
10	Q(SKOLEMGZSH2)	negated_conjecture	
11	¬in(SKOLEMGZSH4,ev(SKOLEMGZSH1,SKOLEMGZSH2)) ∨ ¬W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH3)	(rewrite (resolve 8 9) 10 3 2)	
12	¬in(SKOLEMGZSH4,ev(SKOLEMGZSH1,SKOLEMGZSH2)) ∨ ¬W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH2)	(rewrite (resolve 8 9) 6 4 3 2)	
13	¬W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH2)	(rewrite (resolve 12 1) 10 3 2)	
14	W(SKOLEMGZSH1,SKOLEMGZSH4,SKOLEMGZSH3)	(rewrite 5 13)	
15	¬in(SKOLEMGZSH4,ev(SKOLEMGZSH1,SKOLEMGZSH2))	(rewrite 11 14)	
16	$$FALSE	(rewrite 7 15 13)	

SNARK Proof

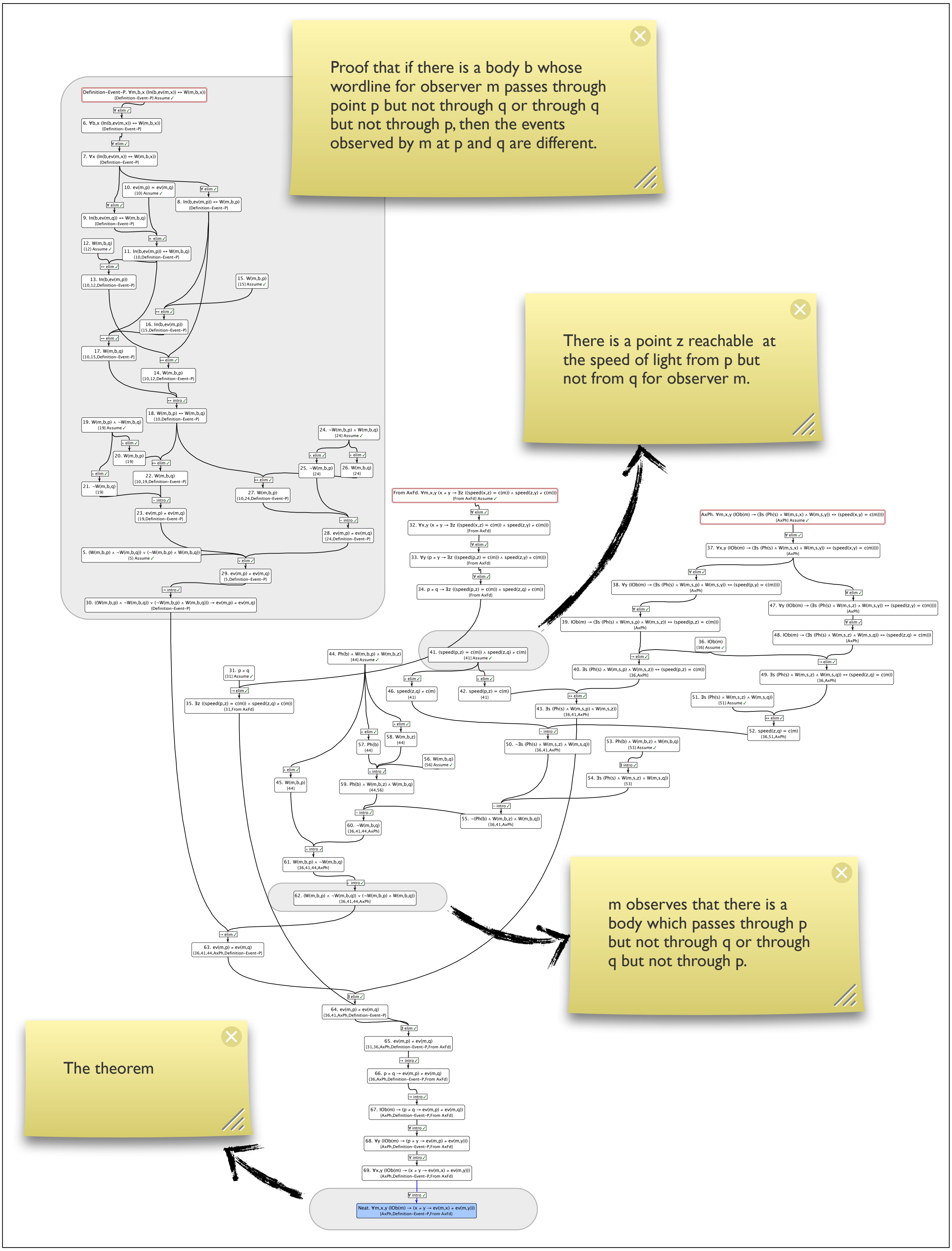
Lin	Formula	Justification	Premise
2	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ (Y = Z) ∨ ¬B(U) ∨ ¬W(X,U,Y) ∨ W(X,U,Z) ∨ ¬(ev(X,Y) = ev(X,Z))	assertion	∀m,p,q ((IOb(m) ∧ Q(p) ∧ Q(q)) → (p ≠ q → (∃b (B(b) ∧ ((W(m,b,p) ∧ ¬
3	ev(SKOLEMJTII2,SKOLEMJTII3) = ev(SKOLEMJTII2,SKOLEMJTII4)	negated_conjecture	
4	¬(SKOLEMJTII3 = SKOLEMJTII4)	negated_conjecture	
5	Q(SKOLEMJTII3)	negated_conjecture	
6	Q(SKOLEMJTII4)	negated_conjecture	
7	IOb(SKOLEMJTII2)	negated_conjecture	
8	¬B(X) ∨ ¬W(SKOLEMJTII2,X,SKOLEMJTII4) ∨ W(SKOLEMJTII2,X,SKOLEMJTII3)	(rewrite (resolve 2 3) 4 5 6 7)	
9	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ (Y = Z) ∨ W(X,SKOLEMJTII1(X,Y,Z),Y) ∨ W(X,SKOLEMJTII1(X,Y,Z),Z)	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ∃b (B(b) ∧ ((W(m,b,x) ∧ ¬W
10	W(SKOLEMJTII2,SKOLEMJTII1(SKOLEMJTII2,SKOLEMJTII4,SKOLEMJTII3),SKOLEMJTII4) ∨ W(SKOLEMJTII2,SKOLEMJTII1(SKOLEMJTII2,SKOLEMJTII4,SKOLEMJTII3),SKOLEMJTII3)	(rewrite (hyperresolve 9 7 6 5) 4)	
11	¬IOb(X) ∨ ¬Q(Y) ∨ ¬Q(Z) ∨ (Y = Z) ∨ B(SKOLEMJTII1(X,Y,Z))	assertion	∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ∃b (B(b) ∧ ((W(m,b,x) ∧ ¬W
12	B(SKOLEMJTII1(SKOLEMJTII2,SKOLEMJTII4,SKOLEMJTII3))	(rewrite (hyperresolve 11 7 6 5) 4)	
13	W(SKOLEMJTII2,SKOLEMJTII1(SKOLEMJTII2,SKOLEMJTII4,SKOLEMJTII3),SKOLEMJTII3)	(hyperresolve 8 10 12)	
14	¬B(X) ∨ ¬W(SKOLEMJTII2,X,SKOLEMJTII3) ∨ W(SKOLEMJTII2,X,SKOLEMJTII4)	(rewrite (resolve 2 3) 4 6 5 7)	
15	¬W(SKOLEMJTII2,SKOLEMJTII1(SKOLEMJTII2,SKOLEMJTII4,SKOLEMJTII3),SKOLEMJTII3) ∨ W(SKOLEMJTII2,SKOLEMJTII1(SKOLEMJTII2,SKOLEMJTII4,SKOLEMJTII3),SKOLEMJTII4)	(resolve 14 12)	
16	W(SKOLEMJTII2,SKOLEMJTII1(SKOLEMJTII2,SKOLEMJTII4,SKOLEMJTII3),SKOLEMJTII4)	(rewrite 15 13)	
17	$$FALSE	(rewrite (resolve 1 13) 16 4 5 6 7)	

Figure 2: Resolution and Paramodulation Slate Proofs of Neat

it. At the conference, we will report upon progress made in attacking, from our AI-based approach, field-theoretic reasoning from the axioms for Neat and NFTLIO.

5.2 Visual Reasoning

Reasoning in physics in general tends to be greatly hybrid in nature; e.g., such reasoning mixes geometrical and visual reasoning that meshes content such as Figure 4 and Figure 5 (from (2) and (3)) with informal semantic reasoning represented using English and informal math. An ideal formal system that can do justice to the way physicists reason should not only incorporate natural-deduction style reasoning as in Slate, but also have provisions for visual proofs and justifications. Some initial work has been done in this regard has been carried out via the using Vivid system introduced in (5; 8), which can combine


Definition–Event–P. ∀m,b,x (In(b,ev(m,x)) ↔ W(m,b,x))
{Definition–Event–P} Assume ✓
∀ elim ✓
6. ∀b,x (In(b,ev(m,x)) ↔ W(m,b,x))
{Definition–Event–P}
∀ elim ✓
7. ∀x (In(b,ev(m,x)) ↔ W(m,b,x))
{Definition–Event–P}
10. ev(m,p) = ev(m,q)
{10} Assume ✓
∀ elim ✓
8. In(b,ev(m,p)) ↔ W(m,b,p)
{Definition–Event–P}
∀ elim ✓
9. In(b,ev(m,q)) ↔ W(m,b,q)
{Definition–Event–P}
= elim ✓
12. W(m,b,q)
{12} Assume ✓
11. In(b,ev(m,p)) ↔ W(m,b,q)
{10,Definition–Event–P}
↔ elim ✓
13. In(b,ev(m,p))
{10,12,Definition–Event–P}
15. W(m,b,p)
{15} Assume ✓
↔ elim ✓
16. In(b,ev(m,p))
{15,Definition–Event–P}
↔ elim ✓
17. W(m,b,q)
{10,15,Definition–Event–P}
↔ elim ✓
14. W(m,b,p)
{10,12,Definition–Event–P}
↔ intro ✓
18. W(m,b,p) ↔ W(m,b,q)
{10,Definition–Event–P}
19. W(m,b,p) ∧ ¬W(m,b,q)
{19} Assume ✓
∧ elim ✓
20. W(m,b,p)
{19}
∧ elim ✓
↔ elim ✓
21. ¬W(m,b,q)
{19}
22. W(m,b,q)
{10,19,Definition–Event–P}
¬ intro ✓
23. ev(m,p) ≠ ev(m,q)
{19,Definition–Event–P}
24. ¬W(m,b,p) ∧ W(m,b,q)
{24} Assume ✓
∧ elim ✓
∧ elim ✓
25. ¬W(m,b,p)
{24}
26. W(m,b,q)
{24}
↔ elim ✓
27. W(m,b,p)
{10,24,Definition–Event–P}
¬ intro ✓
28. ev(m,p) ≠ ev(m,q)
{24,Definition–Event–P}
5. (W(m,b,p) ∧ ¬W(m,b,q)) ∨ (¬W(m,b,p) ∧ W(m,b,q))
{5} Assume ✓
∨ elim ✓
29. ev(m,p) ≠ ev(m,q)
{5,Definition–Event–P}
→ intro ✓
30. ((W(m,b,p) ∧ ¬W(m,b,q)) ∨ (¬W(m,b,p) ∧ W(m,b,q))) → ev(m,p) ≠ ev(m,q)
{Definition–Event–P}
From AxFd. ∀m,x,y (x ≠ y → ∃z ((speed(x,z) = c(m)) ∧ speed(z,y) ≠ c(m)))
{From AxFd} Assume ✓
∀ elim ✓
32. ∀x,y (x ≠ y → ∃z ((speed(x,z) = c(m)) ∧ speed(z,y) ≠ c(m)))
{From AxFd}
∀ elim ✓
33. ∀y (p ≠ y → ∃z ((speed(p,z) = c(m)) ∧ speed(z,y) ≠ c(m)))
{From AxFd}
∀ elim ✓
34. p ≠ q → ∃z ((speed(p,z) = c(m)) ∧ speed(z,q) ≠ c(m))
{From AxFd}
AxPh. ∀m,x,y (IOb(m) → (∃s (Ph(s) ∧ W(m,s,x) ∧ W(m,s,y)) ↔ (speed(x,y) = c(m))))
{AxPh} Assume ✓
∀ elim ✓
37. ∀x,y (IOb(m) → (∃s (Ph(s) ∧ W(m,s,x) ∧ W(m,s,y)) ↔ (speed(x,y) = c(m))))
{AxPh}
∀ elim ✓
38. ∀y (IOb(m) → (∃s (Ph(s) ∧ W(m,s,p) ∧ W(m,s,y)) ↔ (speed(p,y) = c(m))))
{AxPh}
∀ elim ✓
47. ∀y (IOb(m) → (∃s (Ph(s) ∧ W(m,s,z) ∧ W(m,s,y)) ↔ (speed(z,y) = c(m))))
{AxPh}
∀ elim ✓
39. IOb(m) → (∃s (Ph(s) ∧ W(m,s,p) ∧ W(m,s,z)) ↔ (speed(p,z) = c(m)))
{AxPh}
∀ elim ✓
48. IOb(m) → (∃s (Ph(s) ∧ W(m,s,z) ∧ W(m,s,q)) ↔ (speed(z,q) = c(m)))
{AxPh}
36. IOb(m)
{36} Assume ✓
→ elim ✓
40. ∃s (Ph(s) ∧ W(m,s,p) ∧ W(m,s,z)) ↔ (speed(p,z) = c(m))
{36,AxPh}
→ elim ✓
49. ∃s (Ph(s) ∧ W(m,s,z) ∧ W(m,s,q)) ↔ (speed(z,q) = c(m))
{36,AxPh}
44. Ph(b) ∧ W(m,b,p) ∧ W(m,b,z)
{44} Assume ✓
41. (speed(p,z) = c(m)) ∧ speed(z,q) ≠ c(m)
{41} Assume ✓
31. p ≠ q
{31} Assume ✓
→ elim ✓
35. ∃z ((speed(p,z) = c(m)) ∧ speed(z,q) ≠ c(m))
{31,From AxFd}
∧ elim ✓
∧ elim ✓
46. speed(z,q) ≠ c(m)
{41}
42. speed(p,z) = c(m)
{41}
↔ elim ✓
43. ∃s (Ph(s) ∧ W(m,s,p) ∧ W(m,s,z))
{36,41,AxPh}
51. ∃s (Ph(s) ∧ W(m,s,z) ∧ W(m,s,q))
{51} Assume ✓
↔ elim ✓
52. speed(z,q) = c(m)
{36,51,AxPh}
∧ elim ✓
∧ elim ✓
57. Ph(b)
{44}
58. W(m,b,z)
{44}
¬ intro ✓
50. ¬∃s (Ph(s) ∧ W(m,s,z) ∧ W(m,s,q))
{36,41,AxPh}
53. Ph(b) ∧ W(m,b,z) ∧ W(m,b,q)
{53} Assume ✓
∃ intro ✓
54. ∃s (Ph(s) ∧ W(m,s,z) ∧ W(m,s,q))
{53}
56. W(m,b,q)
{56} Assume ✓
∧ elim ✓
∧ intro ✓
45. W(m,b,p)
{44}
59. Ph(b) ∧ W(m,b,z) ∧ W(m,b,q)
{44,56}
¬ intro ✓
55. ¬(Ph(b) ∧ W(m,b,z) ∧ W(m,b,q))
{36,41,AxPh}
¬ intro ✓
60. ¬W(m,b,q)
{36,41,44,AxPh}
∧ intro ✓
61. W(m,b,p) ∧ ¬W(m,b,q)
{36,41,44,AxPh}
∨ intro ✓
62. (W(m,b,p) ∧ ¬W(m,b,q)) ∨ (¬W(m,b,p) ∧ W(m,b,q))
{36,41,44,AxPh}
→ elim ✓
63. ev(m,p) ≠ ev(m,q)
{36,41,44,AxPh,Definition–Event–P}
∃ elim ✓
64. ev(m,p) ≠ ev(m,q)
{36,41,AxPh,Definition–Event–P}
∃ elim ✓
65. ev(m,p) ≠ ev(m,q)
{31,36,AxPh,Definition–Event–P,From AxFd}
→ intro ✓
66. p ≠ q → ev(m,p) ≠ ev(m,q)
{36,AxPh,Definition–Event–P,From AxFd}
→ intro ✓
67. IOb(m) → (p ≠ q → ev(m,p) ≠ ev(m,q))
{AxPh,Definition–Event–P,From AxFd}
∀ intro ✓
68. ∀y (IOb(m) → (p ≠ y → ev(m,p) ≠ ev(m,y)))
{AxPh,Definition–Event–P,From AxFd}
∀ intro ✓
69. ∀x,y (IOb(m) → (x ≠ y → ev(m,x) ≠ ev(m,y)))
{AxPh,Definition–Event–P,From AxFd}
∀ intro ✓
Neat. ∀m,x,y (IOb(m) → (x ≠ y → ev(m,x) ≠ ev(m,y)))
{AxPh,Definition–Event–P,From AxFd}

Proof that if there is a body b whose wordline for observer m passes through point p but not through q or through q but not through p, then the events observed by m at p and q are different.

There is a point z reachable at the speed of light from p but not from q for observer m.

m observes that there is a body which passes through p but not through q or through q but not through p.

The theorem

Figure 3: Manual Proof in Slate of Neat

diagrammatic and symbolic reasoning.[5]

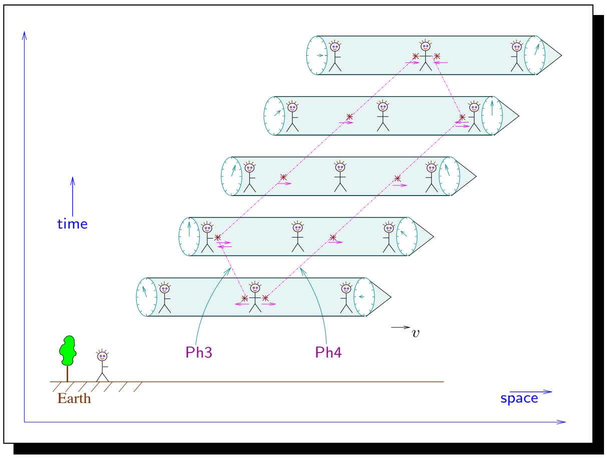


Figure 4: Conceptual Reasoning Using Diagrams

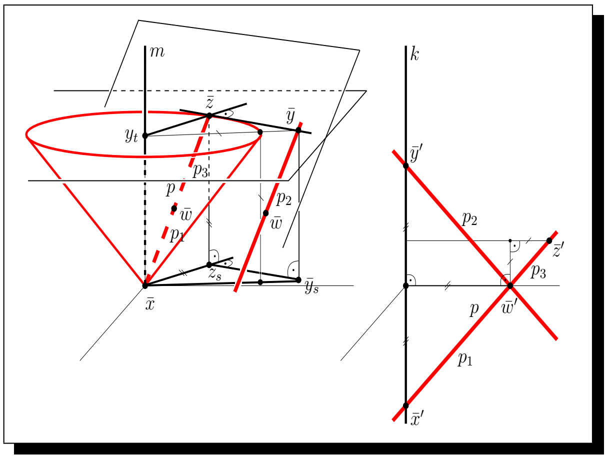


Figure 5: Geometric Reasoning Using Diagrams

References

[1] Hajnal Andréka, Judit X. Madarász, and István Németi. *On the logical structure of relativity theories*. Alfréd Rényi Institute of Mathematics, 2002. With contributions from A. Andai, G. Sági, I. Sain, and Cs. Töke. http://www.math-inst.hu/pub/algebraic-logic/olsort.html.

[2] Hajnal Andréka, Judit X. Madarász, and István Németi. Logic of space-time and relativity theory. In Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem, editors, *Handbook of Spatial Logics*, pages 607–711. Springer, 2007.

[3] Hajnal Andréka, Judit X. Madarász, István Németi, and Gergely Székely. A logic road from special relativity to general relativity. *Synthese*, pages 1–17, 2011.

[4] K. Arkoudas. *Denotational Proof Languages*. PhD thesis, MIT, 2000.

[5] K. Arkoudas. Combining diagrammatic and symbolic reasoning. Technical Report 2005-59, MIT Computer Science and Artificial Intelligence Lab, Cambridge, USA, October 2005.

[6] K. Arkoudas and S. Bringsjord. Propositional Attitudes and Causation. *International Journal of Software and Informatics*, 3(1):47–65, 2009.

[7] Konstantine Arkoudas and Selmer Bringsjord. Computers, justification, and mathematical knowledge. *Minds and Machines*, 17(2):185–202, 2007.

[8] Konstantine Arkoudas and Selmer Bringsjord. Vivid: An AI framework for heterogeneous problem solving. *Artificial Intelligence*, 173(15):1367–1405, 2009. The url http://kryten.mm.rpi.edu/vivid/vivid.pdf provides a preprint of the penultimate draft only. If for some reason it is not working, please contact either author directly by email.

[9] J. Barwise and J. Etchemendy. Heterogeneous logic. In J. Glasgow, N.H. Narayanan, and B. Chandrasekaran, editors, *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, pages 211–234. MIT Press, Cambridge, MA, 1995.

[5] The criterion for distinguishing diagrammatic representations from symbolic representations is that former are *homomorphic* in some fashion to what is being represented; see (9) for a discussion of this form of representation, versus symbolic representations. The distinction is discussed in the context of logic-based AI and computational logic generally in (8).

[10] G. S. Boolos, J. P. Burgess, and R. C. Jeffrey. *Computability and Logic (Fourth Edition)*. Cambridge University Press, Cambridge, UK, 2003.

[11] Selmer Bringsjord, Joshua Taylor, Andrew Shilliday, Micah Clark, and Konstantine Arkoudas. Slate: An Argument-Centered Intelligent Assistant to Human Reasoners. In Floriana Grasso, Nancy Green, Rodger Kibble, and Chris Reed, editors, *Proceedings of the 8th International Workshop on Computational Models of Natural Argument (CMNA 8)*, pages 1–10, Patras, Greece, July 21 2008.

[12] C. Glymour. *Thinking Things Through*. MIT Press, Cambridge, MA, 1992.

[13] Judit X. Madarász. *Logic and Relativity (in the light of definability theory)*. PhD thesis, Eötvös Loránd University, Budapest, Hungary, 2002.

[14] Adam Naumowicz and Artur Korniłowicz. A brief overview of mizar. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 67–72. Springer Berlin / Heidelberg, 2009.

[15] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, The OBI Consortium, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H Scheuermann, Nigam Shah, Patricia L Whetzel, and Suzanna Lewis. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.

[16] Mark E. Stickel. Snark - sri's new automated reasoning kit snark. http://www.ai.sri.com/~stickel/snark.html, 2008.

[17] Gergely Székely. *First-Order Logic Investigation of Relativity Theory with an Emphasis on Accelerated Observers)*. PhD thesis, Eötvös Loránd University, Budapest, Hungary, 2009.

[18] Joseph Henry Woodger and Alfred Tarski. *The axiomatic method in biology*. The University Press, 1937.