Toward Automated Diagrammatic Reasoning in Axiomatic Physics

Atriya Sen, Selmer Bringsjord, Nick Marton, John Licato

Rensselaer Artificial Intelligence and Reasoning Laboratory

Hierarchy of Ethical Reasoning

$\mathcal{DCEC}^*_{CL}$ + diagrammatic reasoning

$\mathcal{DCEC}^*$ + diagrammatic reasoning

$\mathcal{ADR}^M$ + diagrammatic reasoning

$\mathcal{U}$ (UIMA/Watson-inspired)

DIARC

Hard, Harder, Hardest...

Hard: Proof Verification

Example

- Four Colour Theorem (1976, 2005)[1]
- Flyspeck Project (2014)
    - Verifies proof of the Kepler Conjecture
    - Uses HOL Light and Isabelle assistants

[1] Arkoudas, K. & Bringsjord, S. (2007) "Computers, Justification, and Mathematical Knowledge" Minds and Machines 17.2: 185-202.

Computers, Justification, and Mathematical Knowledge

Konstantine Arkoudas · Selmer Bringsjord

Published online: 23 June 2007

Abstract The original proof of the four-color theorem by Appel and Haken sparked a controversy when Tymoczko used it to argue that the justification provided by unsurveyable proofs carried out by computers cannot be a priori. It also created a lingering impression to the effect that such proofs depend heavily for their soundness on large amounts of computation-intensive custom-built software. Contra Tymoczko, we argue that the justification provided by certain computerized mathematical proofs is not fundamentally different from that provided by surveyable proofs, and can be sensibly regarded as a priori. We also show that the aforementioned impression is mistaken because it fails to distinguish between proof search (the context of discovery) and proof checking (the context of justification). By using mechanized proof assistants capable of producing certificates that can be independently checked, it is possible to carry out complex proofs without the need to trust arbitrary custom-written code. We only need to trust one fixed, small, and simple piece of software: the proof checker. This is not only possible in principle, but is in fact becoming a viable methodology for performing complicated mathematical reasoning. This is evinced by a new proof of the four-color theorem that appeared in 2005, and which was developed and checked in its entirety by a mechanical proof system.

Keywords A priori · Justification · Proofs · Certificates · Four-color theorem · Mathematical knowledge

K. Arkoudas (✉) · S. Bringsjord
Cognitive Science Department, Computer Science Department, RPI, Troy, NY, USA 12180
e-mail: arkouk@rpi.edu

S. Bringsjord
e-mail: brings@rpi.edu

Harder: Proof Discovery

Examples (first proofs)

- Semi-automated
  - Gödel's First[1] Incompleteness Theorem (2013, RAIRL)
- Fully automated
  - Robbins Conjecture in Otter (1997)

[1] Licato, John, et al. "Analogico-deductive generation of Gödel's first incompleteness theorem from the liar paradox." Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. AAAI Press, 2013.

Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Analogico-Deductive Generation of Gödel's First Incompleteness Theorem from the Liar Paradox*

John Licato, Naveen Sundar Govindarajulu, Selmer Bringsjord, Michael Pomeranz, Logan Gittelson
Rensselaer Polytechnic Institute (RPI)
Troy NY USA
{licatj,govinn,selmer,pomerm,gittel}@rpi.edu

Abstract

Gödel's proof of his famous first incompleteness theorem (**G1**) has quite understandably long been a tantalizing target for those wanting to engineer impressively intelligent computational systems. After all, in establishing **G1**, Gödel did something that by any metric must be classified as stunningly intelligent. We observe that it has long been understood that there is some sort of analogical relationship between the Liar Paradox (**LP**) and **G1**, and that Gödel himself appreciated and exploited the relationship. Yet the exact nature of the relationship has hitherto not been uncovered, by which we mean that the following question has not been answered: Given a description of **LP**, and the suspicion that it may somehow be used by a suitably programmed computing machine to find a proof of the incompleteness of Peano Arithmetic, can such a machine, provided this description as input, produce as output a complete and verifiably correct proof of **G1**? In this paper, we summarize engineering that entails an affirmative answer to this question. Our approach uses what we call *analogico-deductive reasoning* (ADR), which combines analogical and deductive reasoning to produce a full deductive proof of **G1** from **LP**. Our engineering uses a form of ADR based on our META-R system, and a connection between the Liar Sentence in **LP** and Gödel's Fixed Point Lemma, from which **G1** follows quickly.

1 Introduction

Gödel's proofs of his incompleteness theorems are among the greatest intellectual achievements of the 20th century. Even armed with the suggestion that the Liar Paradox (**LP**) might somehow serve as a guide to proving the incompleteness of Peano Arithmetic (**PA**),[1] the level of creativity and philosophical clarity required to actually tie the two concepts together and produce a valid proof is staggering; it certainly should not be controversial to claim that no computational reasoning system can, at present, achieve this sort of feat without significant human assistance.

1.1 Automating the Proof of G1

Prior work devoted to producing computational systems able to prove **G1** have yielded systems that manage to prove this theorem only when the distance between this result and the starting point is quite small. This for example holds for the first (and certainly seminal) foray; i.e., for [?], as explained in [?], where it's shown that the proof of **G1**, because the set of premises includes an ingenious human-devised encoding scheme, is very easy—to the point of being at the level of proofs requested from students in introductory mathematical logic classes.

Likewise, [?] is an exact parallel of the human-devised proof given by [?]. Finally, in much more recent and truly impressive work by [?], there is a move to natural-deduction formats, which we applaud—but the machine essentially begins its processing at a point exceedingly close to where it needs to end up. As Sieg and Field concede: "As axioms we take for granted the representability and derivability conditions for the central syntactic notions as well as the diagonal lemma for constructing self-referential sentences." If one takes for granted such things, finding a proof of **G1** is effortless for a computing machine.[2] In sum, while a lot of commendable work has been done to build the foundation for our prospective work, the daunting formal and engineering challenge of producing a computational system able to produce **G1** without clever seeding from a human remains entirely unmet.

2 The Analogico-Deductive Approach

2.1 Conjecture Generation

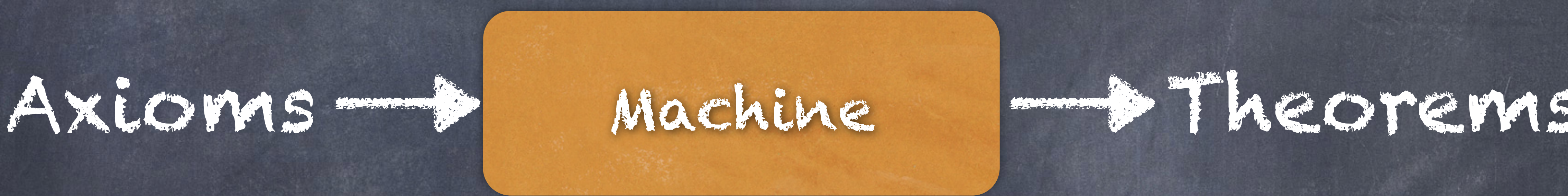
The problem with the purely deductive method is simply that it does not allow us to come close to the type of model-based reasoning that great thinkers are known to have used. Gödel himself has been described as having a "line

*We are deeply grateful for penetrating feedback provided by three anonymous referees, and for financial support from AFOSR and the John Templeton Foundation.

[1] **G1** of course applies to any axiom system meeting the standard conditions (Turing-decidability, representability, consistency), but we tend to refer to **PA** for economization.

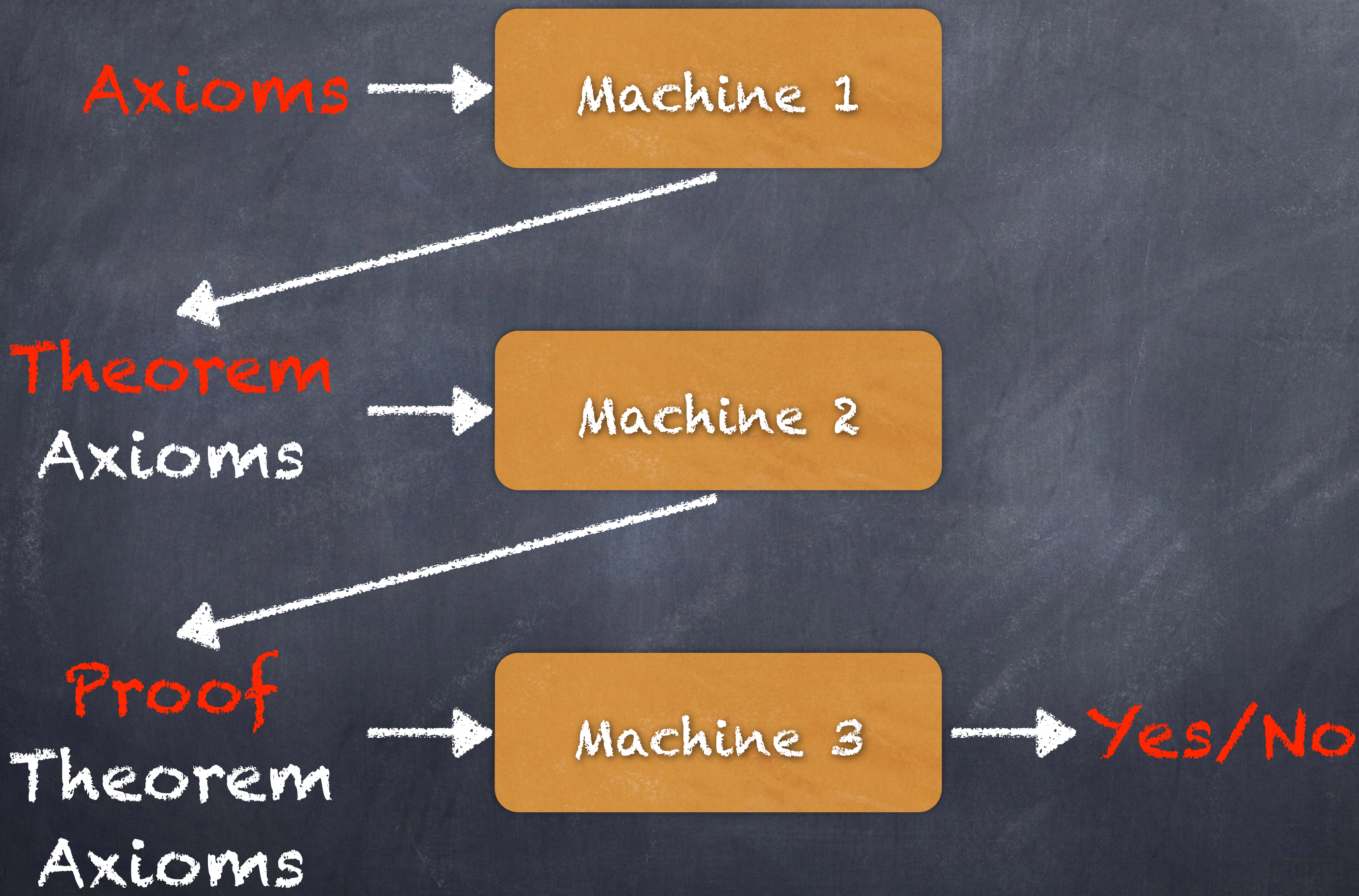
[2] A video demonstration of the small-distance process can be found at http://kryten.mm.rpi.edu/GodelI_abstract_in_Slate.mov.

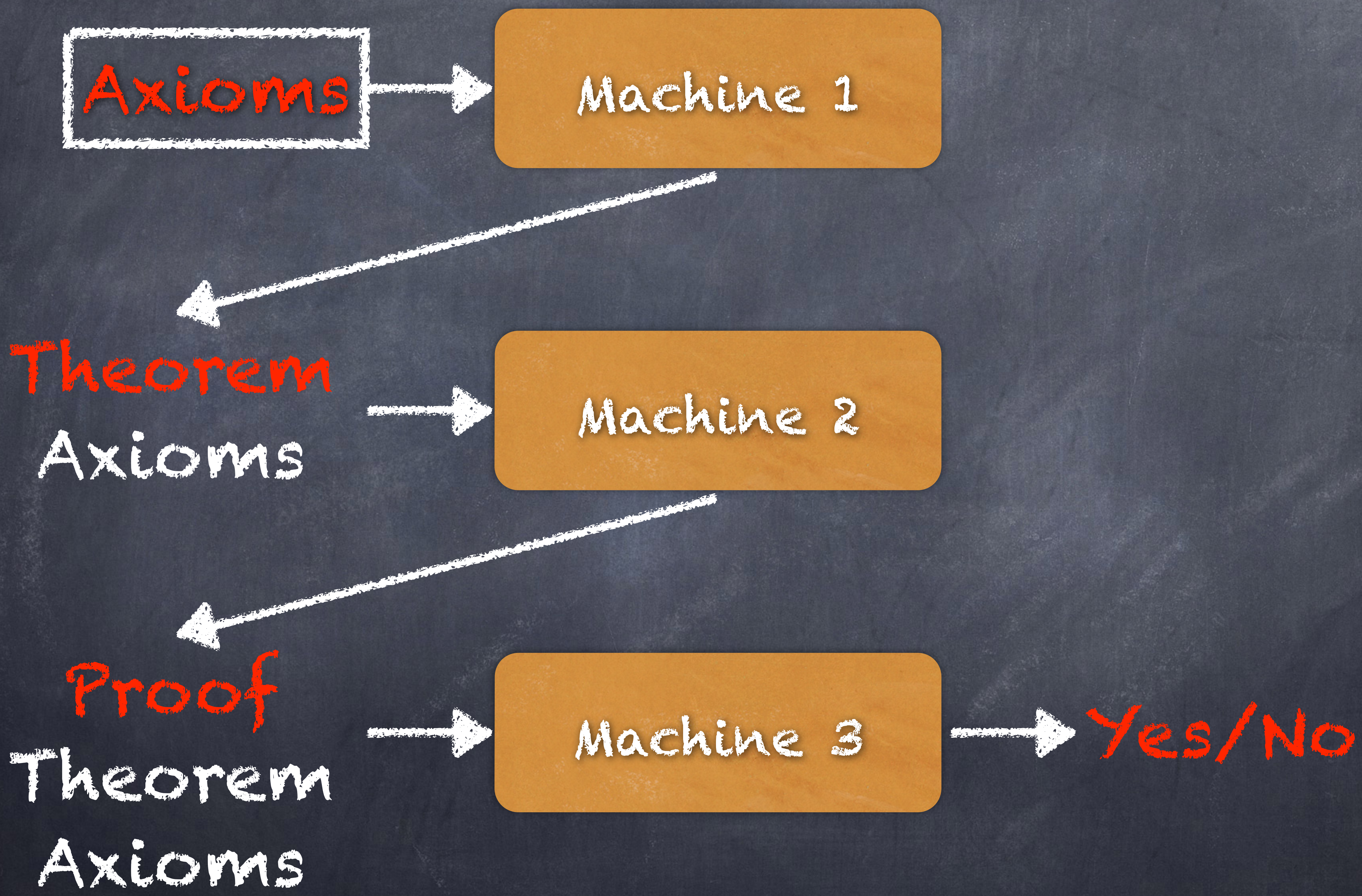
Hardest: Theorem Discovery



Examples

- :(

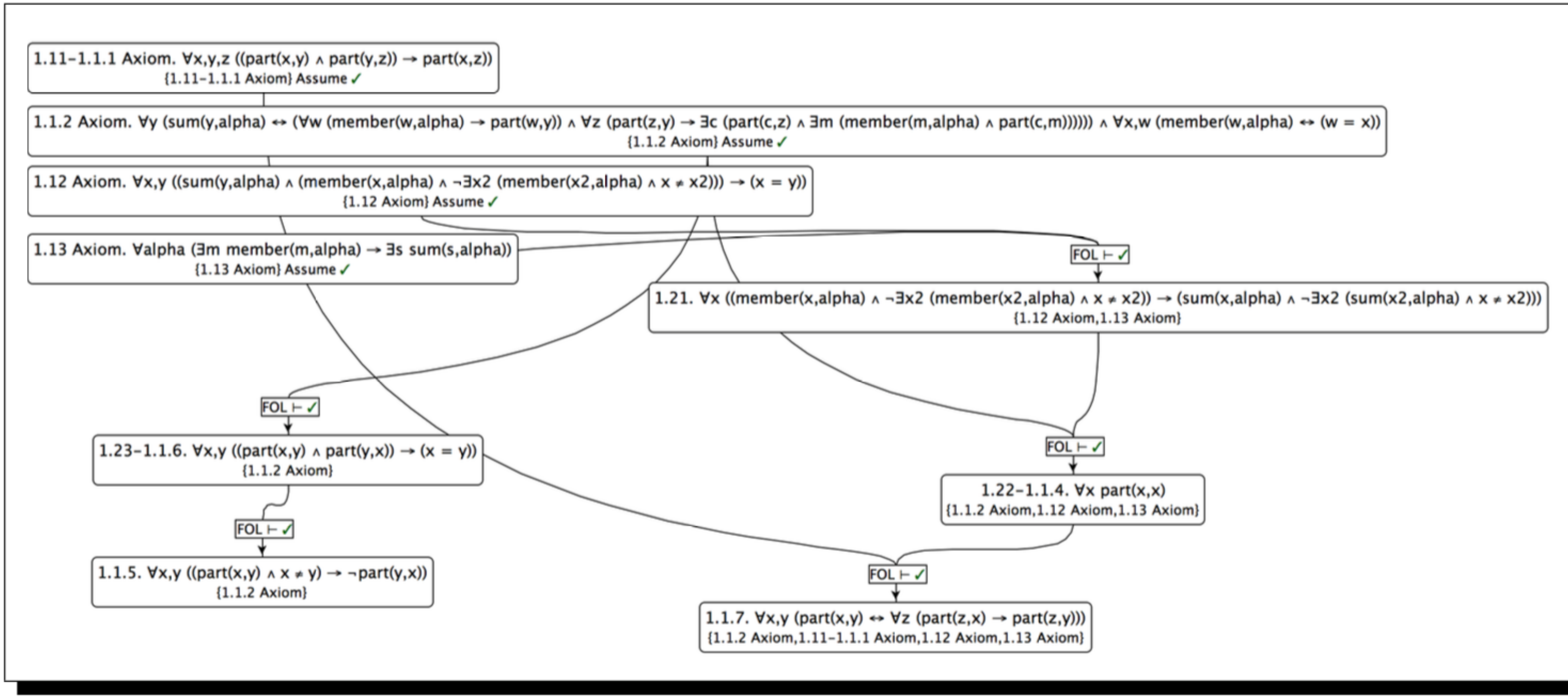

Axioms
Machine 1
Theorem
Axioms
Machine 2
Proof
Theorem
Axioms
Machine 3
Yes/No



Axioms
Machine 1
Theorem
Axioms
Machine 2
Proof
Theorem
Axioms
Machine 3
Yes/No

(Formal) Axiomatizations in the Natural Sciences

- Physics
  - Einstein's Theory of Relativity, due to Andréka et al.
- Biology
  - Mendelian genetics of diploid eukaryotes, due to Woodger (1937)
  - Semi-formally: biological and bio-medical ontologies




```
============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 10.
% Level of proof is 4.
% Maximum clause weight is 0.
% Given clauses 0.

1 (all X (prokaryote_1(X) -> original_name_2(X,ladr6) & description_2(X,ladr5) & user_description_2(X,ladr4) & concept2words_2(X,ladr3) & concept2words_2(X,ladr2) & concept2words_2(X,ladr1) &
unicellular_organism_1(X) & -eukaryote_1(X) & positively_charged_region_1(fn_prokaryote_1(X)) & chemiosmosis_1(fn_prokaryote_2(X)) & protein_enzyme_1(fn_prokaryote_3(X)) &
glycoprotein_1(fn_prokaryote_4(X)) & tangible_entity_1(fn_prokaryote_5(X)) & transmembrane_protein_1(fn_prokaryote_5(X)) & integral_protein_1(fn_prokaryote_5(X)) & transport_protein_1(fn_prokaryote_5(X))
& tangible_entity_1(fn_prokaryote_6(X)) & membrane_protein_1(fn_prokaryote_6(X)) & protein_1(fn_prokaryote_6(X)) & tangible_entity_1(fn_prokaryote_7(X)) & membrane_protein_1(fn_prokaryote_7(X)) &
protein_1(fn_prokaryote_7(X)) & spatial_entity_1(fn_prokaryote_7(X)) & polymer_1(fn_prokaryote_7(X)) & ribosome_1(fn_prokaryote_8(X)) & chromosome_1(fn_prokaryote_9(X)) &
prokaryotic_chromosome_1(fn_prokaryote_9(X)) & substance_1(fn_prokaryote_10(X)) & cytosol_1(fn_prokaryote_10(X)) & tangible_entity_1(fn_prokaryote_11(X)) & phospholipid_bilayer_1(fn_prokaryote_11(X)) &
mixture_1(fn_prokaryote_12(X)) & cytoplasm_1(fn_prokaryote_12(X)) & biomembrane_1(fn_prokaryote_13(X)) & plasma_membrane_1(fn_prokaryote_13(X)) & cell_wall_1(fn_prokaryote_14(X)) &
negatively_charged_region_1(fn_prokaryote_15(X)) & surface_1(fn_prokaryote_16(X)) & cytoplasmic_side_1(fn_prokaryote_16(X)) & outside_face_1(fn_prokaryote_17(X)) &
extracellular_side_1(fn_prokaryote_17(X)) & nucleoid_1(fn_prokaryote_18(X)) & is_along_2(fn_prokaryote_17(X),fn_prokaryote_1(X)) & is_along_2(fn_prokaryote_16(X),fn_prokaryote_15(X)) &
is_along_2(fn_prokaryote_15(X),fn_prokaryote_16(X)) & is_along_2(fn_prokaryote_1(X),fn_prokaryote_17(X)) & has_region_2(X,fn_prokaryote_18(X)) & has_part_2(X,fn_prokaryote_3(X)) &
site_2(fn_prokaryote_2(X),fn_prokaryote_13(X)) & is_inside_2(fn_prokaryote_13(X),fn_prokaryote_14(X)) & called_2(fn_prokaryote_6(X),ladr0) & fn_prokaryote_6(X) != fn_prokaryote_5(X) & fn_prokaryote_5(X) !
= fn_prokaryote_6(X) & has_part_2(X,fn_prokaryote_9(X)) & has_part_2(X,fn_prokaryote_14(X)) & has_region_2(fn_prokaryote_12(X),fn_prokaryote_15(X)) & is_inside_2(fn_prokaryote_9(X),fn_prokaryote_18(X)) &
has_part_2(X,fn_prokaryote_12(X)) & is_facing_2(fn_prokaryote_16(X),fn_prokaryote_12(X)) & is_oriented_toward_2(fn_prokaryote_16(X),fn_prokaryote_12(X)) & has_part_2(X,fn_prokaryote_8(X)) &
is_inside_2(fn_prokaryote_8(X),fn_prokaryote_10(X)) & has_part_2(fn_prokaryote_12(X),fn_prokaryote_10(X)) & is_inside_2(fn_prokaryote_12(X),fn_prokaryote_13(X)) & has_part_2(X,fn_prokaryote_13(X)) &
has_part_2(fn_prokaryote_13(X),fn_prokaryote_11(X)) & has_region_2(fn_prokaryote_13(X),fn_prokaryote_16(X)) & is_outside_2(fn_prokaryote_17(X),fn_prokaryote_13(X)) &
has_region_2(fn_prokaryote_13(X),fn_prokaryote_17(X)) & has_part_2(fn_prokaryote_13(X),fn_prokaryote_6(X)) & has_part_2(fn_prokaryote_13(X),fn_prokaryote_4(X)) &
is_inside_2(fn_prokaryote_4(X),fn_prokaryote_11(X)) & is_inside_2(fn_prokaryote_5(X),fn_prokaryote_11(X)) & is_across_2(fn_prokaryote_5(X),fn_prokaryote_13(X)) &
has_part_2(fn_prokaryote_13(X),fn_prokaryote_5(X)))) # label(a620065) # label(axiom) # label(non_clause).  [assumption].
2 (exists M (has_part_2(ind_for_prokaryote_0,M) & ribosome_1(M))) # label(non_clause) # label(goal).  [goal].
3 prokaryote_1(ind_for_prokaryote_0) # label(a620066) # label(axiom).  [assumption].
28 -prokaryote_1(x) | ribosome_1(fn_prokaryote_8(x)) # label(a620065) # label(axiom).  [clausify(1)].
63 -prokaryote_1(x) | has_part_2(x,fn_prokaryote_8(x)) # label(a620065) # label(axiom).  [clausify(1)].
79 -has_part_2(ind_for_prokaryote_0,x) | -ribosome_1(x).  [deny(2)].
83 has_part_2(ind_for_prokaryote_0,fn_prokaryote_8(ind_for_prokaryote_0)).  [resolve(3,a,63,a)].
91 ribosome_1(fn_prokaryote_8(ind_for_prokaryote_0)).  [resolve(3,a,28,a)].
95 -ribosome_1(fn_prokaryote_8(ind_for_prokaryote_0)).  [resolve(83,a,79,a)].
98 $F.  [resolve(95,a,91,a)].

============================== end of proof ==========================
```

LRB12: Sentential proof of Theorem NEAT (No Event at Two Places)[1]

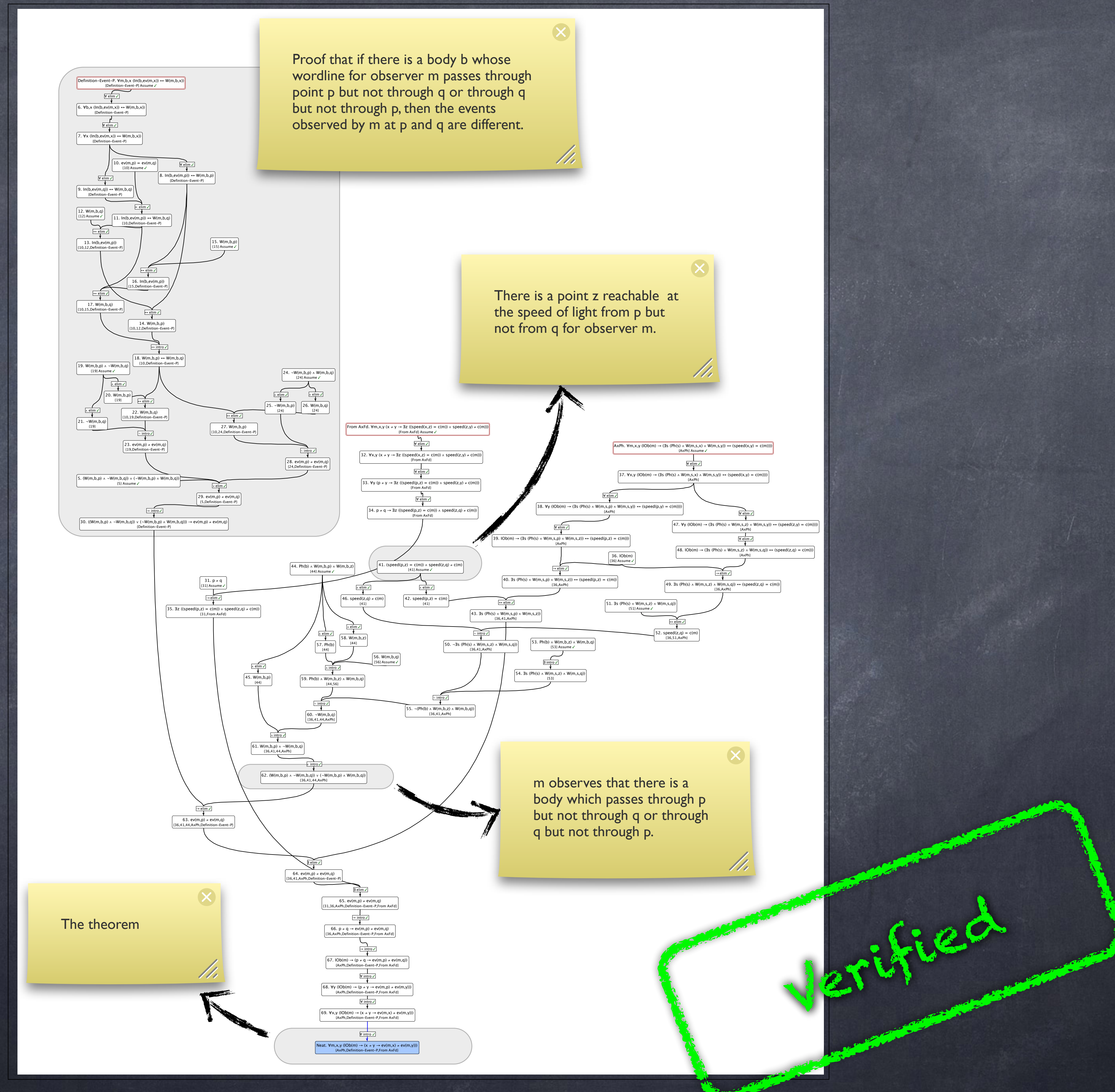
$\forall m,x,y((Ob(m) \wedge Q(x) \wedge Q(y)) \rightarrow (x \neq y \rightarrow ev(m,x) \neq ev(m,y)))$

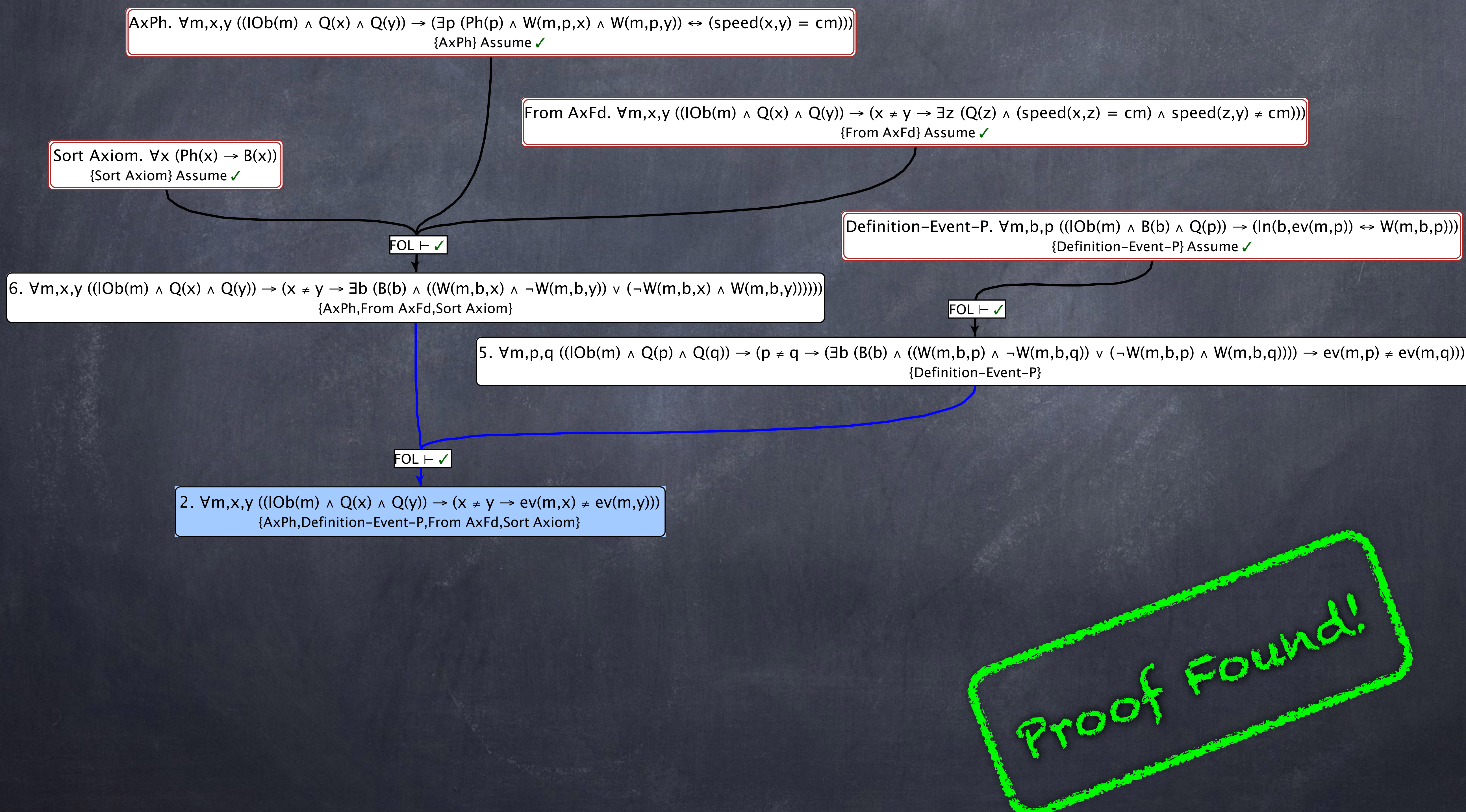
[1] Govindarajalulu, Naveen Sundar, Selmer Bringsjord, and Joshua Taylor. "Proof verification and proof discovery for relativity." Synthese (2014): 1-18.

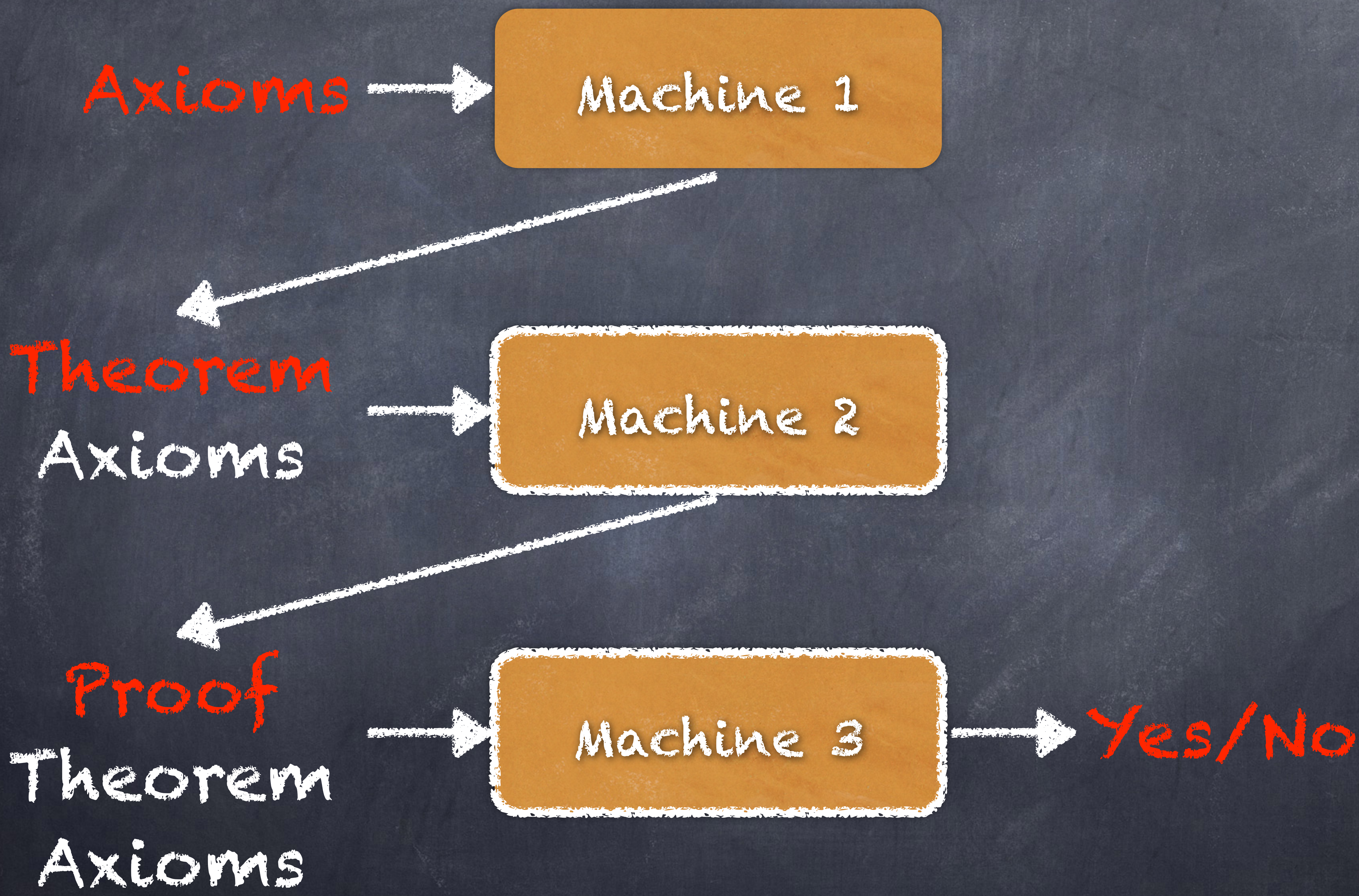
Software

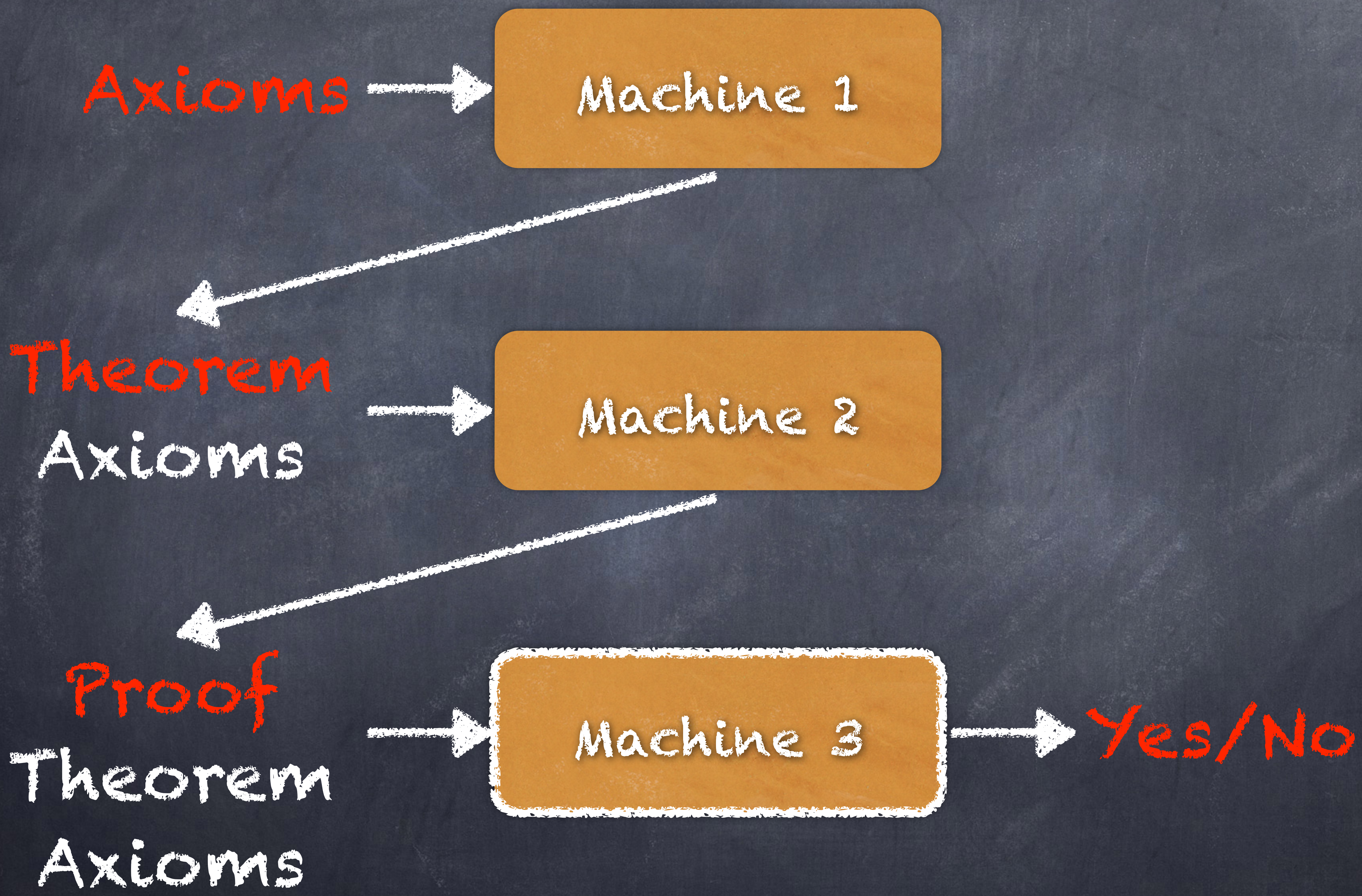
- SNARK: resolution theorem prover for FOL with equality.
- SLATE: Graphical, interactive natural deduction prover, built on SNARK.[1]

[1] Bringsjord, S., Taylor, J., Shilliday, A, Clark, M. & Arkoudas, K. (2008) "Slate: An Argument-Centered Intelligent Assistant to Human Reasoners" in Proceedings of the 8th International Workshop on Computational Models of Natural Argument (CMNA 8), Grasso, F., Green, N., Kibble, R. & Reed, C., eds. Pages 1-10, Patras, Greece, July 21, ISBN: 978-960-6843-12-9.


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
Verified


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 ✓
From AxFd. ∀m,x,y ((IOb(m) ∧ Q(x) ∧ Q(y)) → (x ≠ y → ∃z (Q(z) ∧ (speed(x,z) = cm) ∧ speed(z,y) ≠ cm)))
{From AxFd} Assume ✓
Sort Axiom. ∀x (Ph(x) → B(x))
{Sort Axiom} Assume ✓
FOL ⊢ ✓
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 ✓
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,From AxFd,Sort Axiom}
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,From AxFd,Sort Axiom}
Proof Found!


Axioms
Machine 1
Theorem
Axioms
Machine 2
Proof
Theorem
Axioms
Machine 3
Yes/No


Axioms
Machine 1
Theorem
Axioms
Machine 2
Proof
Theorem
Axioms
Machine 3
Yes/No

Denotational Proof Languages

K. Arkoudas. Denotational Proof Languages. PhD Thesis, MIT, 2000.

Sentence

Proof

$\mathcal{Z}$

$\mathcal{K}$

Logic

DPL

NDL0

- Proof of the tautology A & B ==> B & A:

```
assume A & B
  begin
     left-and A & B;
     right-and A & B;
     both B, A
end
```

$\mathcal{NDL}_1$

- Proof of the tautology: (forall x P(x)) ==> ~ (exists x ~ P(x)):

```
assume (forall x P(x))
  suppose-absurd (exists x ~P(x))
    pick-witness w for (exists x ~P(x)) // we now have
~P(w)
      begin
        P(w) BY specialize (forall x P(x)) with w;
        absurd P(w), ~P(w)
      end.
```

Diagrammatic Reasoning vs. Spatial Logics

Vivid

Konstantine Arkoudas, Selmer Bringsjord, Vivid: A framework for heterogeneous problem solving, Artificial Intelligence, Volume 173, Issue 15, October 2009.

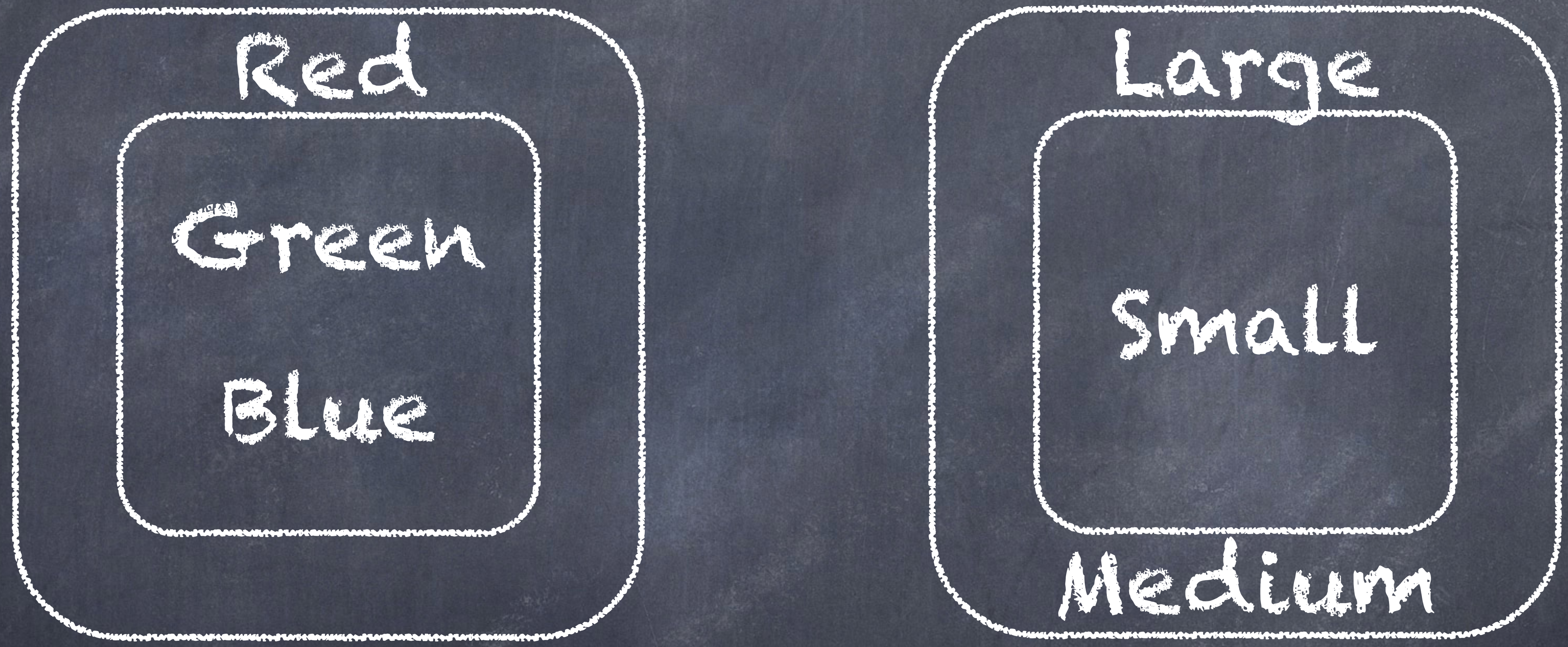
Attribute Structures and Systems

- Consider an attribute system consisting of two clocks c_1 and c_2. Then,

S = ({c_1, c_2}; hours: {0, ..., 23}, minutes: {0, ..., 59})

- Systems have states σ, and certain states are extensions of others. States may also be disjoint.

Alternative State Extensions



Colour of s	Size of s
{red}	{small, large, medium}
{red, green, blue}	{large, medium}

Specifying a Vivid Language

- Attribute Structure $\mathscr{A}$.
- First-order vocabulary $\Sigma = (C, R, V)$.
- Interpretation $\mathscr{I}$ of Σ onto $\mathscr{A}$.

Interpreting First-Order Languages into System States

Attribute Interpretation is a mapping $\mathcal{I}$ that assigns to each relation symbol:

- Realization
- Profile

Running Example: Clock System

- Vocabulary $\Sigma_{clock} = (C_{clock}, R_{clock}, V_{clock})$
  - $C_{clock} = \{c_1, c_2, c_3\}$, $V_{clock} = \{x, y, z, x_1, y_1, z_1\}$, $R_{clock} = \{PM, AM, Ahead, Behind\}$
- Attribute Structure

  Clock = (hours: {0, ..., 23}, minutes: {0, ..., 59}; $\{R_1, R_2, R_3, R_4\}$)
  - $R_1(h) \leftrightarrow h > 11$, and so on
- Interpretation $\mathcal{I}$
  - Realizations: $PM^{\mathcal{I}} = R_1$, and so on
  - Prof(PM) = [(hours, 1)]

Vivid: Syntax

$\mathfrak{D} ::= D \mid \Delta$

- Logical inference rules modus ponens, double negation.
- Sentential deduction rules: assume, pick-any, specialize.
- Diagrammatic deduction rules: thinning, cases.

$$
\begin{array}{lcl}
D & ::= & \mathit{RuleApp} \\
 & | & \textbf{assume}\ F\ \ D \\
 & | & F\ \ \textbf{by}\ \ D \\
 & | & \mathfrak{D}; D \\
 & | & \textbf{pick-any}\ x\ \ D \\
 & | & \textbf{pick-witness}\ w\ \ \textbf{for}\ \exists\, x\, .\, F\ \ D \\
 & | & \textbf{specialize}\ \forall\, x_1 \cdots x_n\, .\, F\ \textbf{with}\ t_1, \ldots, t_n \\
 & | & \textbf{ex-generalize}\ \exists\, x\, .\, F\ \textbf{from}\ t \\
 & | & \textbf{cases by}\ F_1, \ldots, F_k\text{:}\ (\sigma_1; \rho_1) \rightarrow\ D_1 \mid \cdots \mid (\sigma_n; \rho_n) \rightarrow\ D_n \\
 & | & \textbf{observe}\ \ F
\end{array}
$$

$$
\begin{array}{lcl}
\Delta & ::= & \mathfrak{D}; \Delta \\
 & | & \textbf{claim}\ (\sigma; \rho) \\
 & | & (\sigma; \rho)\ \textbf{by thinning with}\ F_1, \ldots, F_n \\
 & | & (\sigma; \rho)\ \textbf{by widening} \\
 & | & (\sigma; \rho)\ \textbf{by absurdity} \\
 & | & \textbf{cases by}\ F_1, \ldots, F_k\text{:}\ (\sigma_1; \rho_1) \rightarrow\ \Delta_1 \mid \cdots \mid (\sigma_n; \rho_n) \rightarrow\ \Delta_n \\
 & | & \textbf{cases}\ F_1\ \vee\ F_2\text{:}\ F_1\ \rightarrow\ \Delta_1 \mid F_2\ \rightarrow\ \Delta_2 \\
 & | & \textbf{pick-witness}\ w\ \ \textbf{for}\ \exists\, x\, .\, F\ \ \Delta
\end{array}
$$

$$
\mathfrak{D}\ \ ::=\ \ D \mid \Delta
$$

Vivid: Evaluation Semantics $\gamma \vdash D \rightarrow F$ and $\gamma \vdash \Delta \rightsquigarrow (\sigma;\rho)$

- In the context γ, deduction $D(\Delta)$ derives F (respectively, $(\sigma;\rho)$)
- Evaluation semantics of Vivid rules

$$\frac{}{(\beta \cup \{F_1, \ldots, F_n\}; (\sigma; \rho)) \vdash (\sigma'; \rho') \textbf{ by thinning with } F_1, \ldots, F_n \leadsto (\sigma'; \rho')} \quad [\textit{Thinning}]$$

provided $(\sigma; \rho) \Vdash_{\{F_1, \ldots, F_n\}} (\sigma'; \rho')$

$$\frac{}{(\beta; (\sigma; \rho)) \vdash (\sigma'; \rho') \textbf{ by widening} \leadsto (\sigma'; \rho')} \quad [\textit{Widening}]$$

provided $(\sigma; \rho) \sqsubseteq (\sigma'; \rho')$

$$\frac{}{(\beta \cup \{\textbf{false}\}; (\sigma; \rho)) \vdash (\sigma'; \rho') \textbf{ by absurdity} \leadsto (\sigma'; \rho')} \quad [\textit{Absurdity}]$$

$$\frac{}{(\beta; (\sigma; \rho)) \vdash \textbf{claim } (\sigma; \rho) \leadsto (\sigma; \rho)} \quad [\textit{Diagram-Reiteration}]$$

$$\frac{\begin{array}{c}(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_1; \rho_1)) \vdash \Delta_1 \leadsto (\sigma'; \rho') \\ \vdots \\ (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_n; \rho_n)) \vdash \Delta_n \leadsto (\sigma'; \rho')\end{array}}{(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \vdash \textbf{cases by } F_1, \ldots, F_k\text{: } (\sigma_1; \rho_1) \rightarrow \Delta_1 \mid \cdots \mid (\sigma_n; \rho_n) \rightarrow \Delta_n \leadsto (\sigma'; \rho')} \quad [C_1]$$

provided $(\sigma; \rho) \Vdash_{\{F_1, \ldots, F_k\}} \{(\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)\}$

$$\frac{(\beta \cup \{F_1 \vee F_2, F_1\}; (\sigma; \rho)) \vdash \Delta_1 \leadsto (\sigma'; \rho') \quad (\beta \cup \{F_1 \vee F_2, F_2\}; (\sigma; \rho)) \vdash \Delta_2 \leadsto (\sigma'; \rho')}{(\beta \cup \{F_1 \vee F_2\}; (\sigma; \rho)) \vdash \textbf{cases } F_1 \vee F_2\text{: } F_1 \rightarrow \Delta_1 \mid F_2 \rightarrow \Delta_2 \leadsto (\sigma'; \rho')} \quad [C_2]$$

$$\frac{(\beta; (\sigma; \rho)) \vdash D \leadsto F \quad (\beta \cup \{F\}; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho')}{(\beta; (\sigma; \rho)) \vdash D; \Delta \leadsto (\sigma'; \rho')} \quad [D; \Delta]$$

$$\frac{(\beta; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho') \quad (\beta; (\sigma'; \rho')) \vdash D \leadsto F}{(\beta; (\sigma; \rho)) \vdash \Delta; D \leadsto F} \quad [\Delta; D]$$

$$\frac{(\beta; (\sigma; \rho)) \vdash \Delta_1 \leadsto (\sigma_1; \rho_1) \quad (\beta; (\sigma_1; \rho_1)) \vdash \Delta_2 \leadsto (\sigma_2; \rho_2)}{(\beta; (\sigma; \rho)) \vdash \Delta_1; \Delta_2 \leadsto (\sigma_2; \rho_2)} \quad [\Delta; \Delta]$$

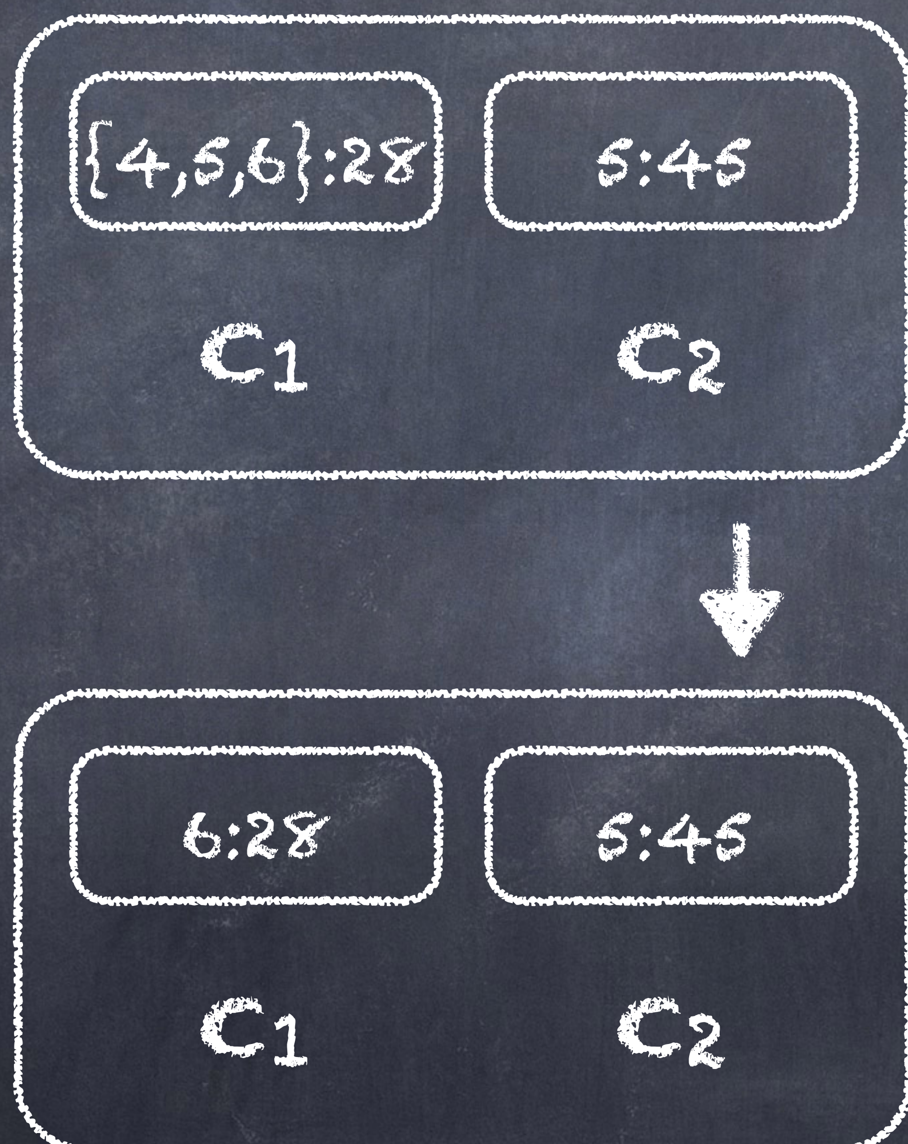
$$\frac{(\beta; (\sigma; \rho)) \vdash D_1 \leadsto F_1 \quad (\beta \cup \{F_1\}; (\sigma; \rho)) \vdash D_2 \leadsto F_2}{(\beta; (\sigma; \rho)) \vdash D_1; D_2 \leadsto F_2} \quad [D; D]$$

$$\frac{(\beta \cup \{\exists\, x \,.\, F, F\,[z/x]\}; (\sigma; \rho)) \vdash \Delta\,[z/w] \leadsto (\sigma'; \rho')}{(\beta \cup \{\exists\, x \,.\, F\}; (\sigma; \rho)) \vdash \textbf{pick-witness } w \textbf{ for } \exists\, x \,.\, F \;\; \Delta \leadsto (\sigma'; \rho')} \quad [\textit{EI}/\Delta]$$

provided z is fresh

Running Example: Clock System

State σ



+ Ahead(c_1, c_2)

State σ'

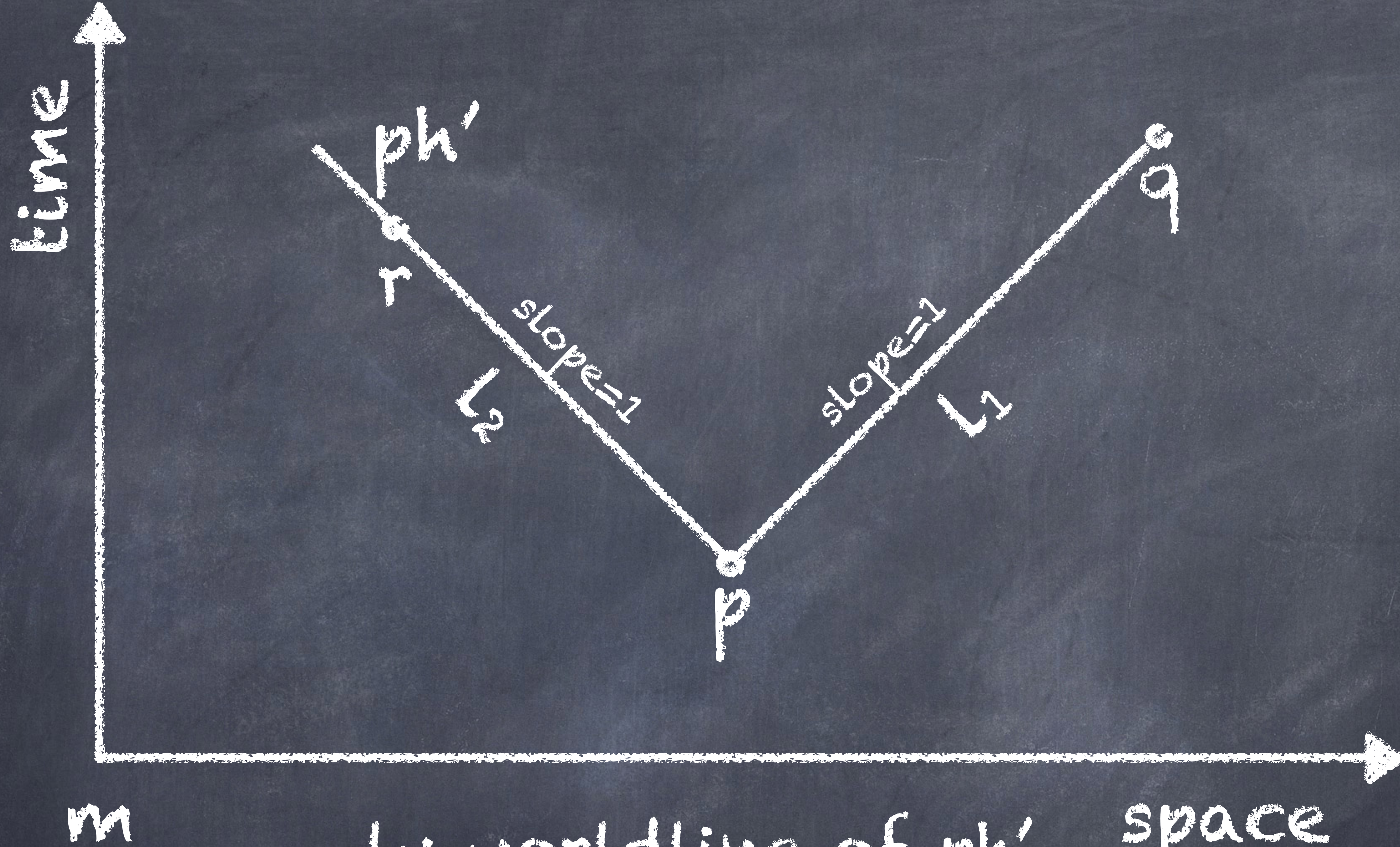
(σ'; ρ) by thinning with Ahead(c_1, c_2)

Diagrammatic Formal Verification of Theorem NEAT

Theorem NEAT/2.2[1]

No observer observes the same event at two different space-time locations.

[1]Andréka, Hajnal, Judit X. Madarász, and István Németi. "Logic of space-time and relativity theory." Handbook of spatial logics. Springer Netherlands, 2007. 607-711.



l_2: worldline of ph′

$ph' \in ev_m(p)$

$ph' \notin ev_m(q)$

∴ Set of bodies observed by m at p and q are not identical.

Specifying Our Vivid Language

- Attribute Structure

$A = (\{position: \mathbb{R}^{n}, slope: [0,1], line\_positions: \mathbb{R}^{n*}\}; R_1, R_2, R_3)$

- Vocabulary $\Sigma = (C, R, V)$

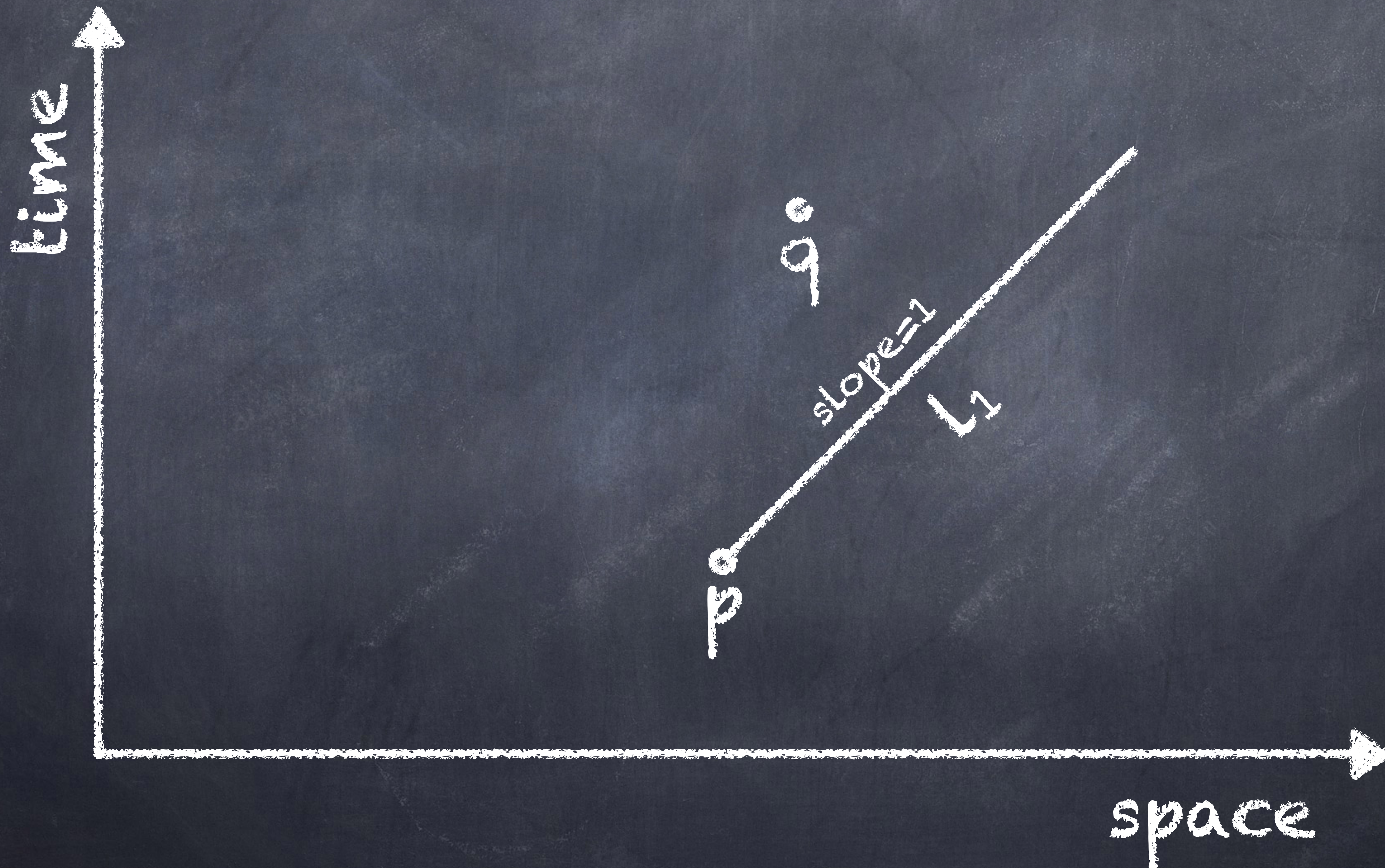
$\Sigma = (\{p, q, m, l_1, l_2\}, \{through(p,l), observes(p,x), slope\_of\_one(l)\}, \varnothing)$

- Interpretation:

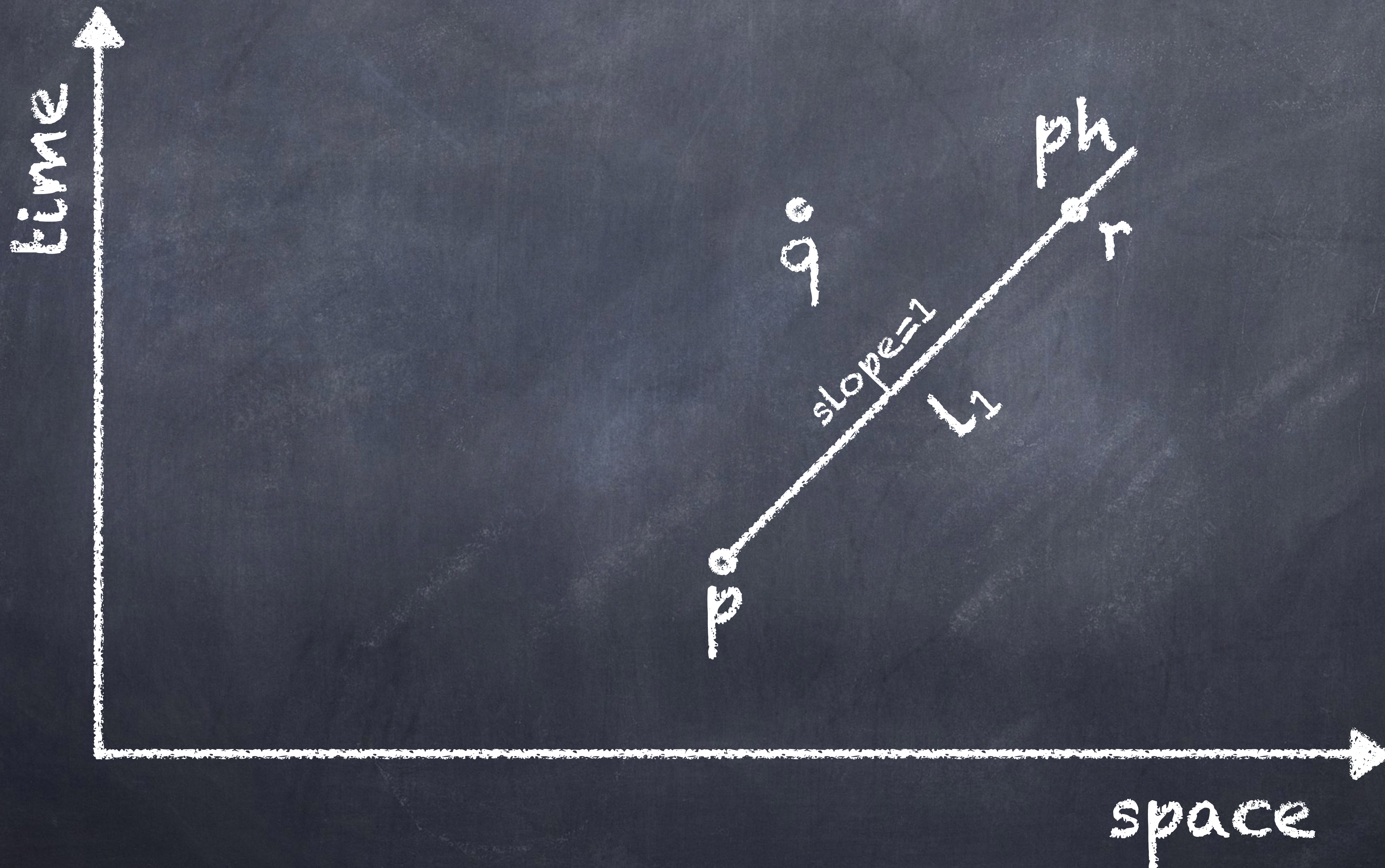
Symbol	Arity	Realization	Profile
through	2	R_1	[(position,1),(line_positions,2)]
observes	2	R_2	[(position,1),(position,2)]
slope_of_one	2	R_3	[(slope,1)]

Proof: Δ_0

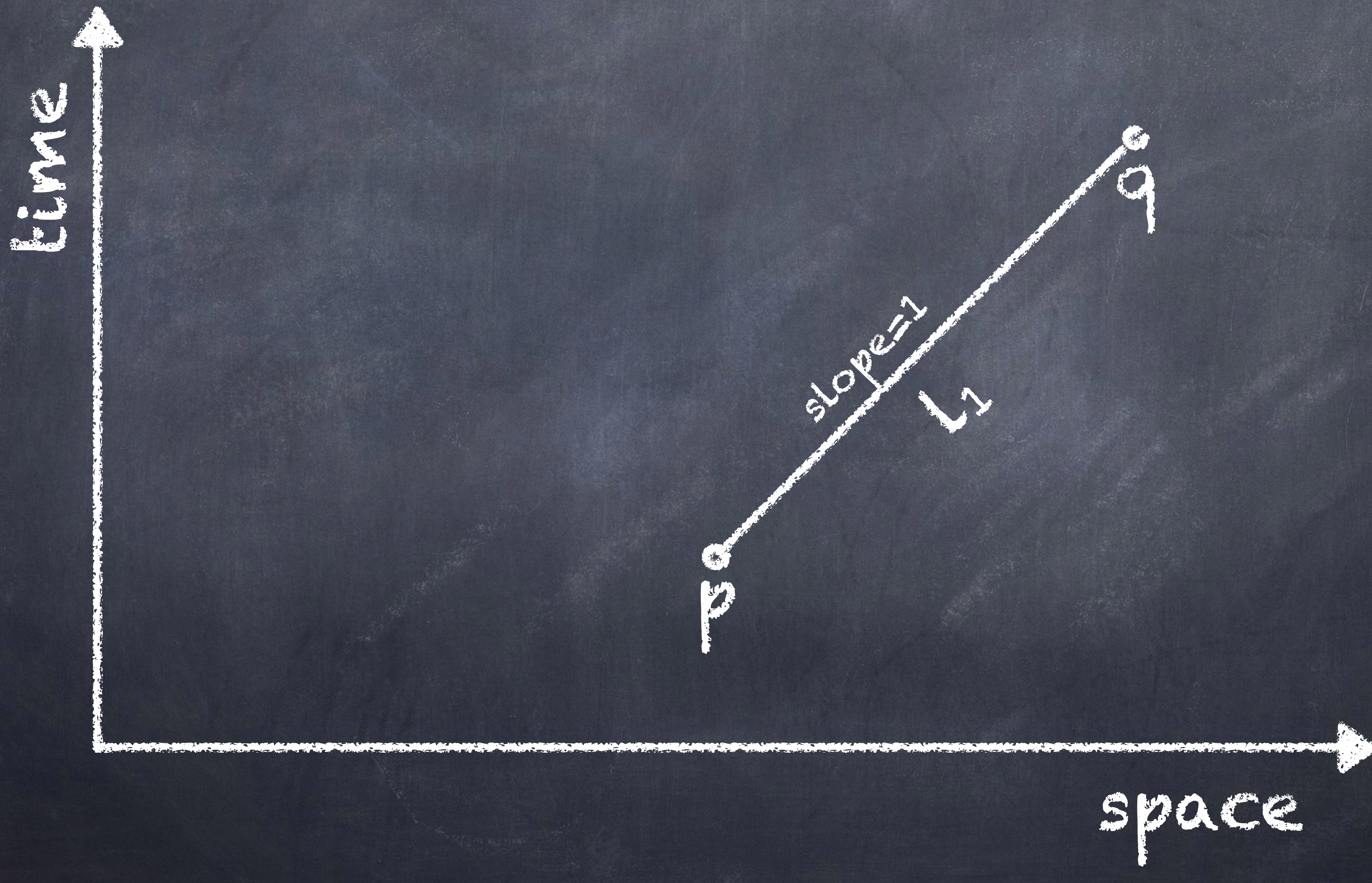
Proof; Case I: Δ_1



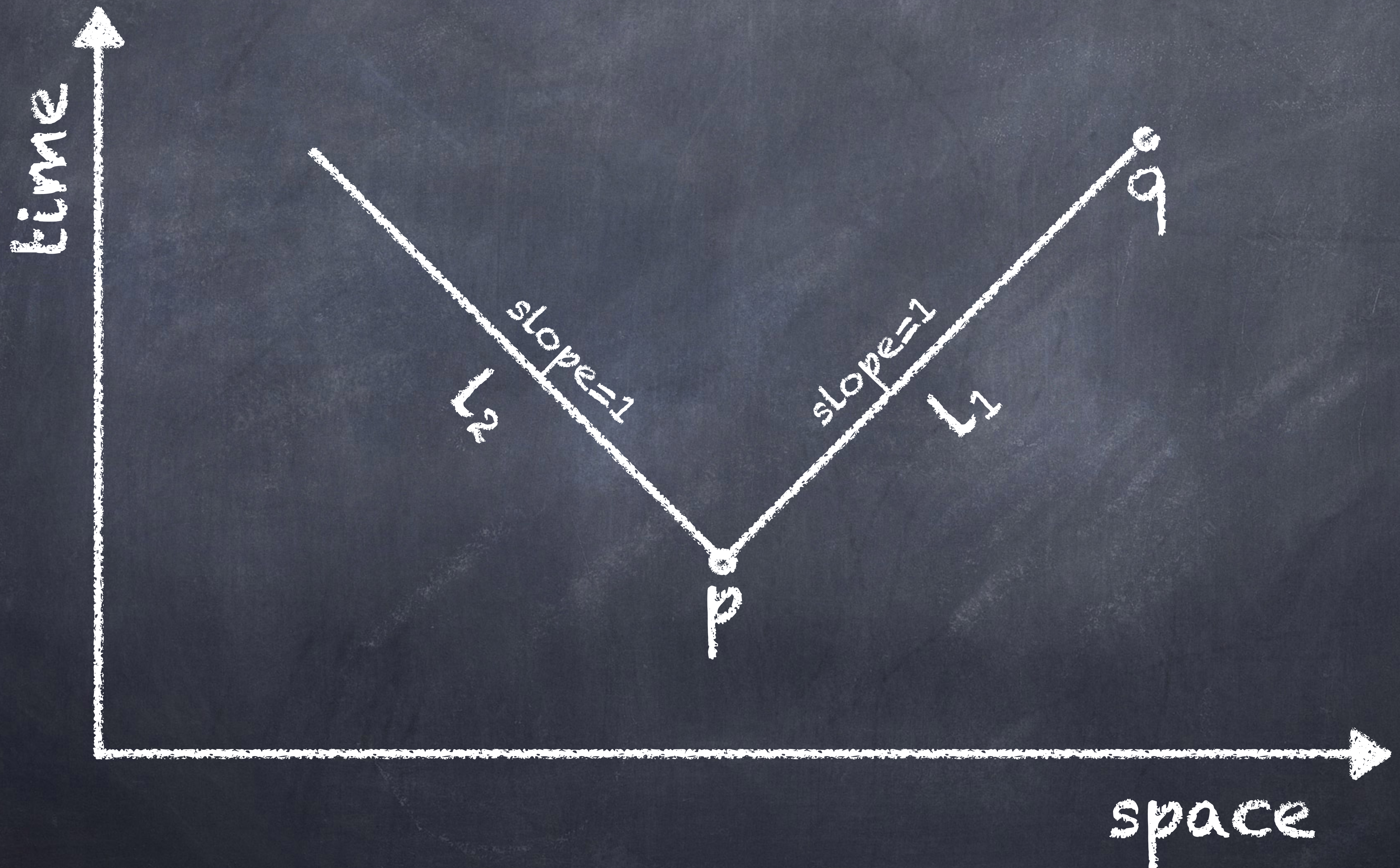
Proof; Case I: Δ_3



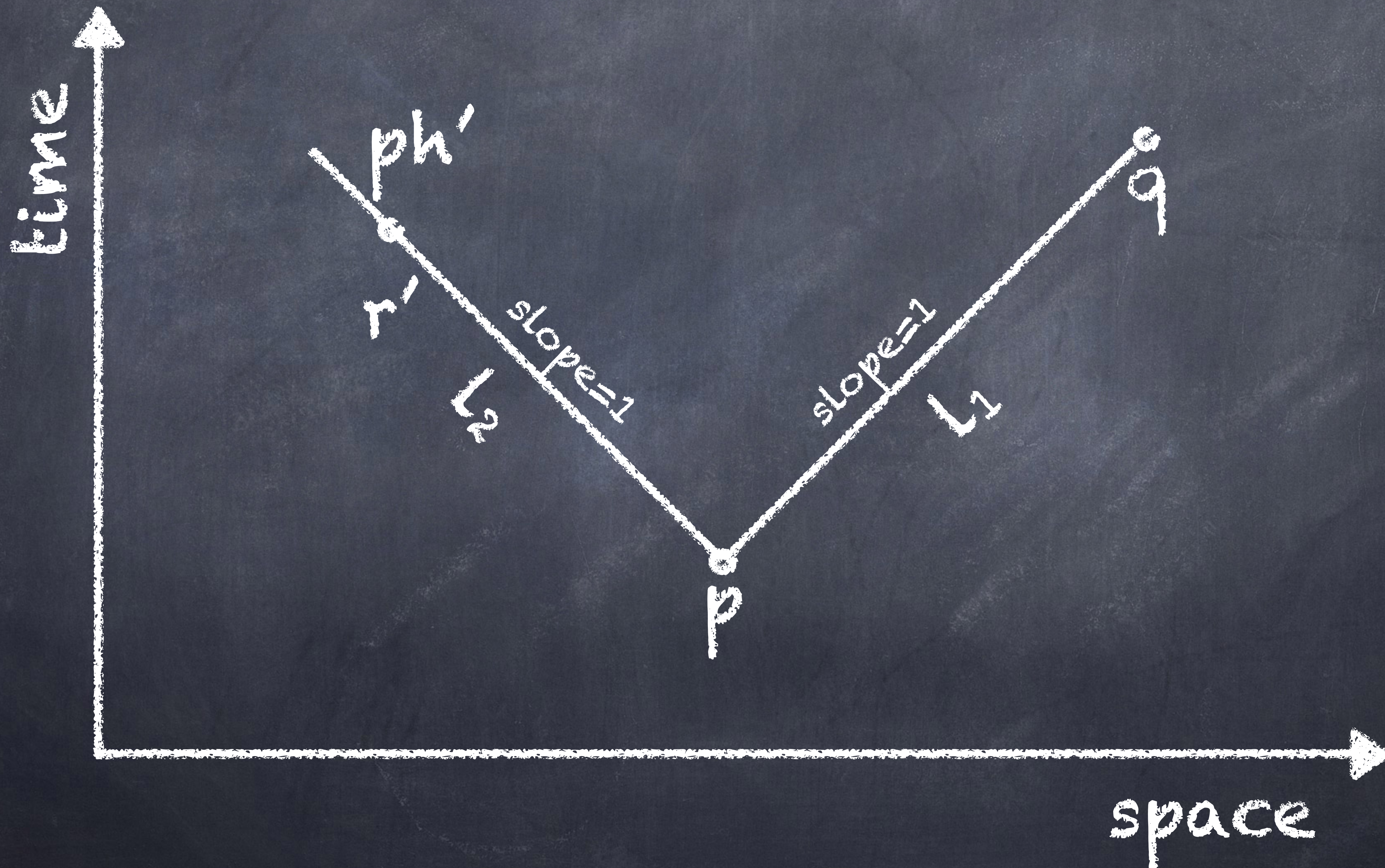
Proof; Case II: Δ_2



Proof; Case II: Δ_4



Proof; Case II: Δ_S



Vivid Proof

from Δ_0 cases by AxFd:

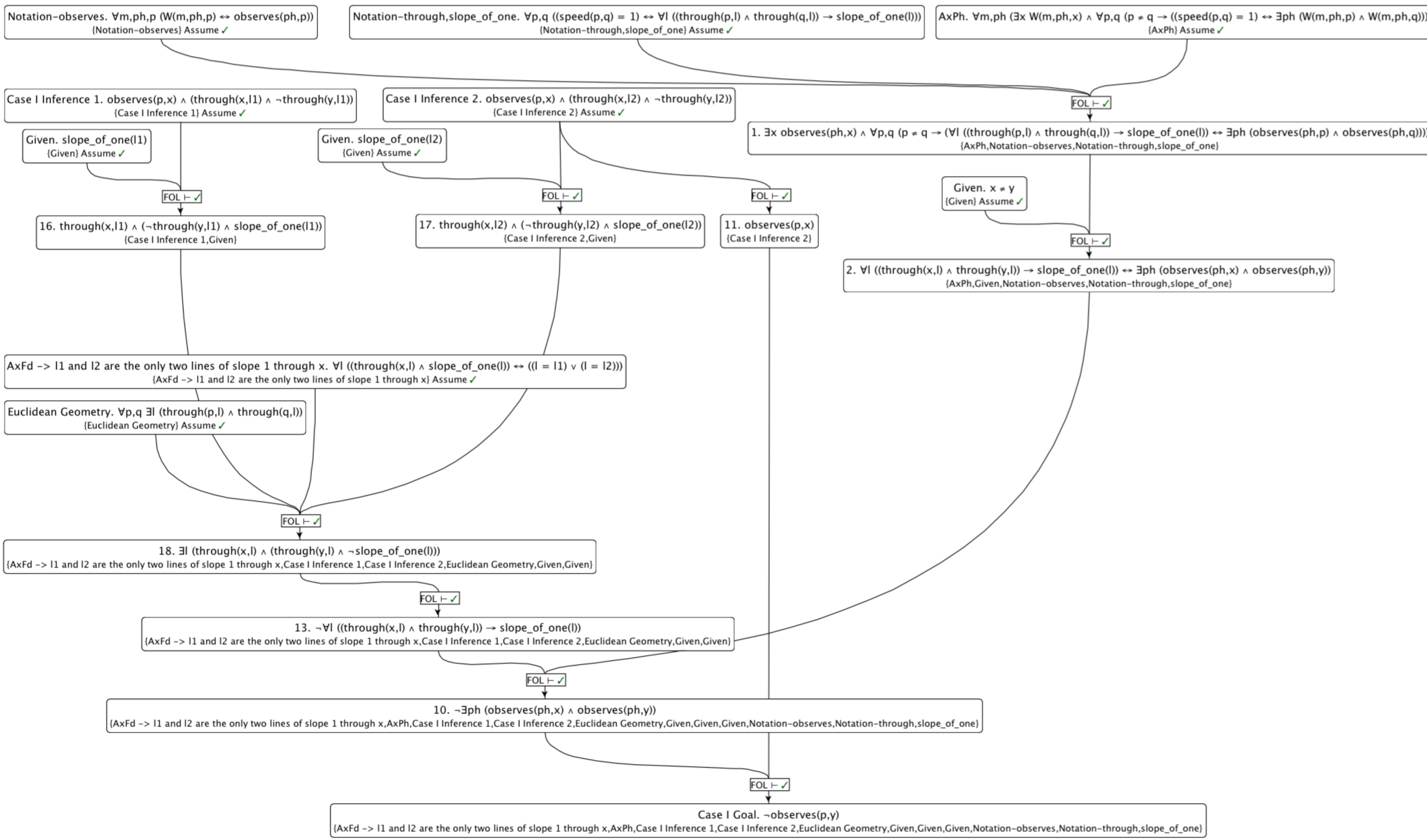
Δ_1 ➔ Δ_3 by claim ph at r

observe observes(ph,r) ∧ through(r,l_1) ∧ through(p,l_1) ∧ ¬through (q,l_1)

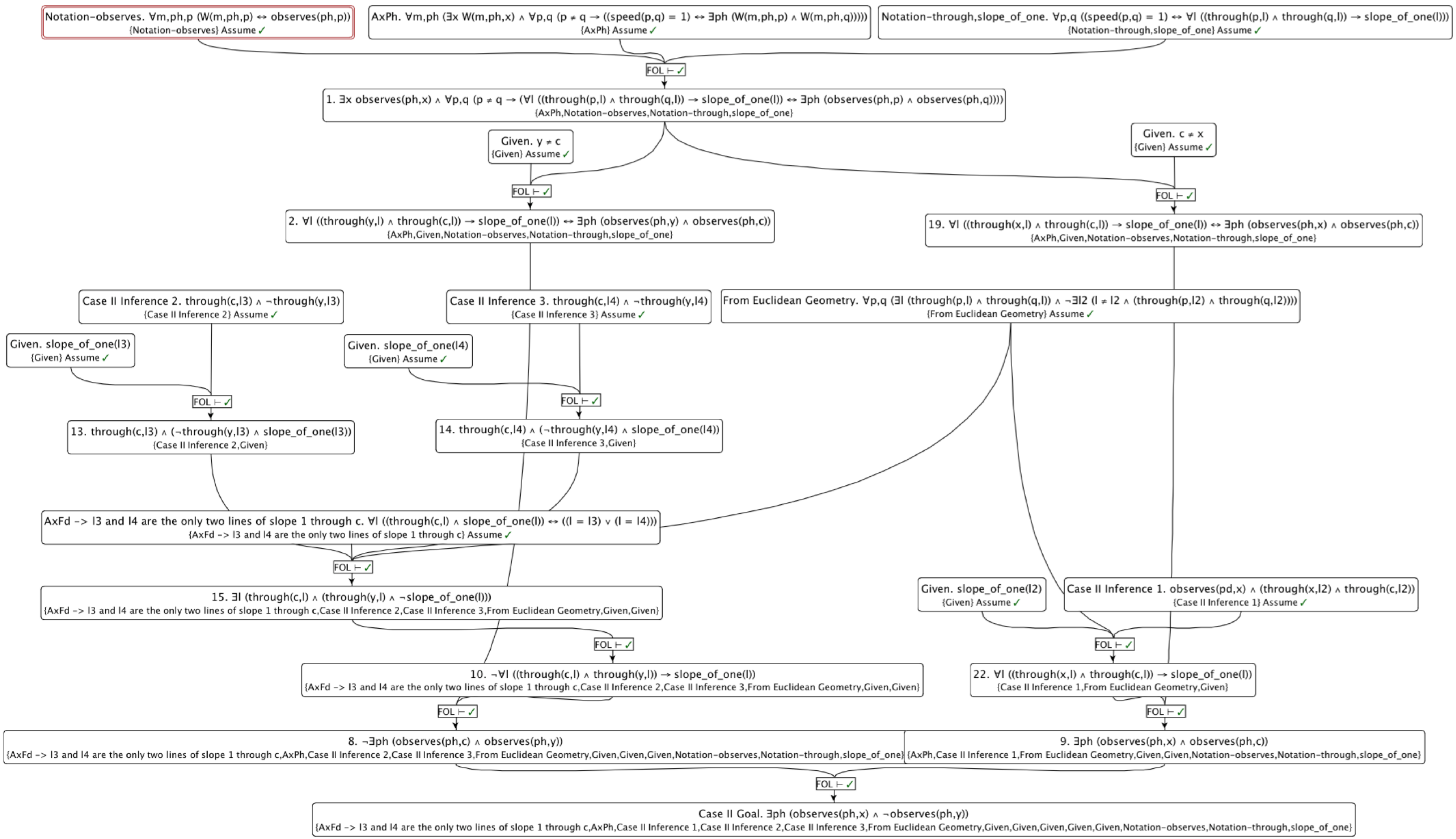
Δ_2 ➔ Δ_4 by D;Δ with AxFd

Δ_5 by claim ph' at r'

observe observes(ph',r') ∧ through(r',l_2) ∧ through(p,l_2) ∧ ¬through (q,l_2)



Notation-observes. ∀m,ph,p (W(m,ph,p) ↔ observes(ph,p))
{Notation-observes} Assume ✓
Notation-through,slope_of_one. ∀p,q ((speed(p,q) = 1) ↔ ∀l ((through(p,l) ∧ through(q,l)) → slope_of_one(l)))
{Notation-through,slope_of_one} Assume ✓
AxPh. ∀m,ph (∃x W(m,ph,x) ∧ ∀p,q (p ≠ q → ((speed(p,q) = 1) ↔ ∃ph (W(m,ph,p) ∧ W(m,ph,q)))))
{AxPh} Assume ✓
Case I Inference 1. observes(p,x) ∧ (through(x,l1) ∧ ¬through(y,l1))
{Case I Inference 1} Assume ✓
Case I Inference 2. observes(p,x) ∧ (through(x,l2) ∧ ¬through(y,l2))
{Case I Inference 2} Assume ✓
FOL ⊢ ✓
1. ∃x observes(ph,x) ∧ ∀p,q (p ≠ q → (∀l ((through(p,l) ∧ through(q,l)) → slope_of_one(l)) ↔ ∃ph (observes(ph,p) ∧ observes(ph,q))))
{AxPh,Notation-observes,Notation-through,slope_of_one}
Given. slope_of_one(l1)
{Given} Assume ✓
Given. slope_of_one(l2)
{Given} Assume ✓
FOL ⊢ ✓
FOL ⊢ ✓
FOL ⊢ ✓
Given. x ≠ y
{Given} Assume ✓
16. through(x,l1) ∧ (¬through(y,l1) ∧ slope_of_one(l1))
{Case I Inference 1,Given}
17. through(x,l2) ∧ (¬through(y,l2) ∧ slope_of_one(l2))
{Case I Inference 2,Given}
11. observes(p,x)
{Case I Inference 2}
FOL ⊢ ✓
2. ∀l ((through(x,l) ∧ through(y,l)) → slope_of_one(l)) ↔ ∃ph (observes(ph,x) ∧ observes(ph,y))
{AxPh,Given,Notation-observes,Notation-through,slope_of_one}
AxFd -> l1 and l2 are the only two lines of slope 1 through x. ∀l ((through(x,l) ∧ slope_of_one(l)) ↔ ((l = l1) ∨ (l = l2)))
{AxFd -> l1 and l2 are the only two lines of slope 1 through x} Assume ✓
Euclidean Geometry. ∀p,q ∃l (through(p,l) ∧ through(q,l))
{Euclidean Geometry} Assume ✓
FOL ⊢ ✓
18. ∃l (through(x,l) ∧ (through(y,l) ∧ ¬slope_of_one(l)))
{AxFd -> l1 and l2 are the only two lines of slope 1 through x,Case I Inference 1,Case I Inference 2,Euclidean Geometry,Given,Given}
FOL ⊢ ✓
13. ¬∀l ((through(x,l) ∧ through(y,l)) → slope_of_one(l))
{AxFd -> l1 and l2 are the only two lines of slope 1 through x,Case I Inference 1,Case I Inference 2,Euclidean Geometry,Given,Given}
FOL ⊢ ✓
10. ¬∃ph (observes(ph,x) ∧ observes(ph,y))
{AxFd -> l1 and l2 are the only two lines of slope 1 through x,AxPh,Case I Inference 1,Case I Inference 2,Euclidean Geometry,Given,Given,Given,Notation-observes,Notation-through,slope_of_one}
FOL ⊢ ✓
Case I Goal. ¬observes(p,y)
{AxFd -> l1 and l2 are the only two lines of slope 1 through x,AxPh,Case I Inference 1,Case I Inference 2,Euclidean Geometry,Given,Given,Given,Notation-observes,Notation-through,slope_of_one}



Notation-observes. ∀m,ph,p (W(m,ph,p) ↔ observes(ph,p))
{Notation-observes} Assume ✓
AxPh. ∀m,ph (∃x W(m,ph,x) ∧ ∀p,q (p ≠ q → ((speed(p,q) = 1) ↔ ∃ph (W(m,ph,p) ∧ W(m,ph,q)))))
{AxPh} Assume ✓
Notation-through,slope_of_one. ∀p,q ((speed(p,q) = 1) ↔ ∀l ((through(p,l) ∧ through(q,l)) → slope_of_one(l)))
{Notation-through,slope_of_one} Assume ✓
FOL ⊢ ✓
1. ∃x observes(ph,x) ∧ ∀p,q (p ≠ q → (∀l ((through(p,l) ∧ through(q,l)) → slope_of_one(l)) ↔ ∃ph (observes(ph,p) ∧ observes(ph,q))))
{AxPh,Notation-observes,Notation-through,slope_of_one}
Given. y ≠ c
{Given} Assume ✓
Given. c ≠ x
{Given} Assume ✓
FOL ⊢ ✓
2. ∀l ((through(y,l) ∧ through(c,l)) → slope_of_one(l)) ↔ ∃ph (observes(ph,y) ∧ observes(ph,c))
{AxPh,Given,Notation-observes,Notation-through,slope_of_one}
FOL ⊢ ✓
19. ∀l ((through(x,l) ∧ through(c,l)) → slope_of_one(l)) ↔ ∃ph (observes(ph,x) ∧ observes(ph,c))
{AxPh,Given,Notation-observes,Notation-through,slope_of_one}
Case II Inference 2. through(c,l3) ∧ ¬through(y,l3)
{Case II Inference 2} Assume ✓
Case II Inference 3. through(c,l4) ∧ ¬through(y,l4)
{Case II Inference 3} Assume ✓
From Euclidean Geometry. ∀p,q (∃l (through(p,l) ∧ through(q,l)) ∧ ¬∃l2 (l ≠ l2 ∧ (through(p,l2) ∧ through(q,l2))))
{From Euclidean Geometry} Assume ✓
Given. slope_of_one(l3)
{Given} Assume ✓
Given. slope_of_one(l4)
{Given} Assume ✓
FOL ⊢ ✓
FOL ⊢ ✓
13. through(c,l3) ∧ (¬through(y,l3) ∧ slope_of_one(l3))
{Case II Inference 2,Given}
14. through(c,l4) ∧ (¬through(y,l4) ∧ slope_of_one(l4))
{Case II Inference 3,Given}
AxFd -> l3 and l4 are the only two lines of slope 1 through c. ∀l ((through(c,l) ∧ slope_of_one(l)) ↔ ((l = l3) ∨ (l = l4)))
{AxFd -> l3 and l4 are the only two lines of slope 1 through c} Assume ✓
FOL ⊢ ✓
15. ∃l (through(c,l) ∧ (through(y,l) ∧ ¬slope_of_one(l)))
{AxFd -> l3 and l4 are the only two lines of slope 1 through c,Case II Inference 2,Case II Inference 3,From Euclidean Geometry,Given,Given}
Given. slope_of_one(l2)
{Given} Assume ✓
Case II Inference 1. observes(pd,x) ∧ (through(x,l2) ∧ through(c,l2))
{Case II Inference 1} Assume ✓
FOL ⊢ ✓
FOL ⊢ ✓
10. ¬∀l ((through(c,l) ∧ through(y,l)) → slope_of_one(l))
{AxFd -> l3 and l4 are the only two lines of slope 1 through c,Case II Inference 2,Case II Inference 3,From Euclidean Geometry,Given,Given}
22. ∀l ((through(x,l) ∧ through(c,l)) → slope_of_one(l))
{Case II Inference 1,From Euclidean Geometry,Given}
FOL ⊢ ✓
FOL ⊢ ✓
8. ¬∃ph (observes(ph,c) ∧ observes(ph,y))
{AxFd -> l3 and l4 are the only two lines of slope 1 through c,AxPh,Case II Inference 2,Case II Inference 3,From Euclidean Geometry,Given,Given,Given,Notation-observes,Notation-through,slope_of_one}
9. ∃ph (observes(ph,x) ∧ observes(ph,c))
{AxPh,Case II Inference 1,From Euclidean Geometry,Given,Given,Notation-observes,Notation-through,slope_of_one}
FOL ⊢ ✓
Case II Goal. ∃ph (observes(ph,x) ∧ ¬observes(ph,y))
{AxFd -> l3 and l4 are the only two lines of slope 1 through c,AxPh,Case II Inference 1,Case II Inference 2,Case II Inference 3,From Euclidean Geometry,Given,Given,Given,Given,Given,Notation-observes,Notation-through,slope_of_one}

Vivid Proof

from Δ_0 cases by AxFd:

$\Delta_1 \rightarrow \Delta_3$ by claim ph at r

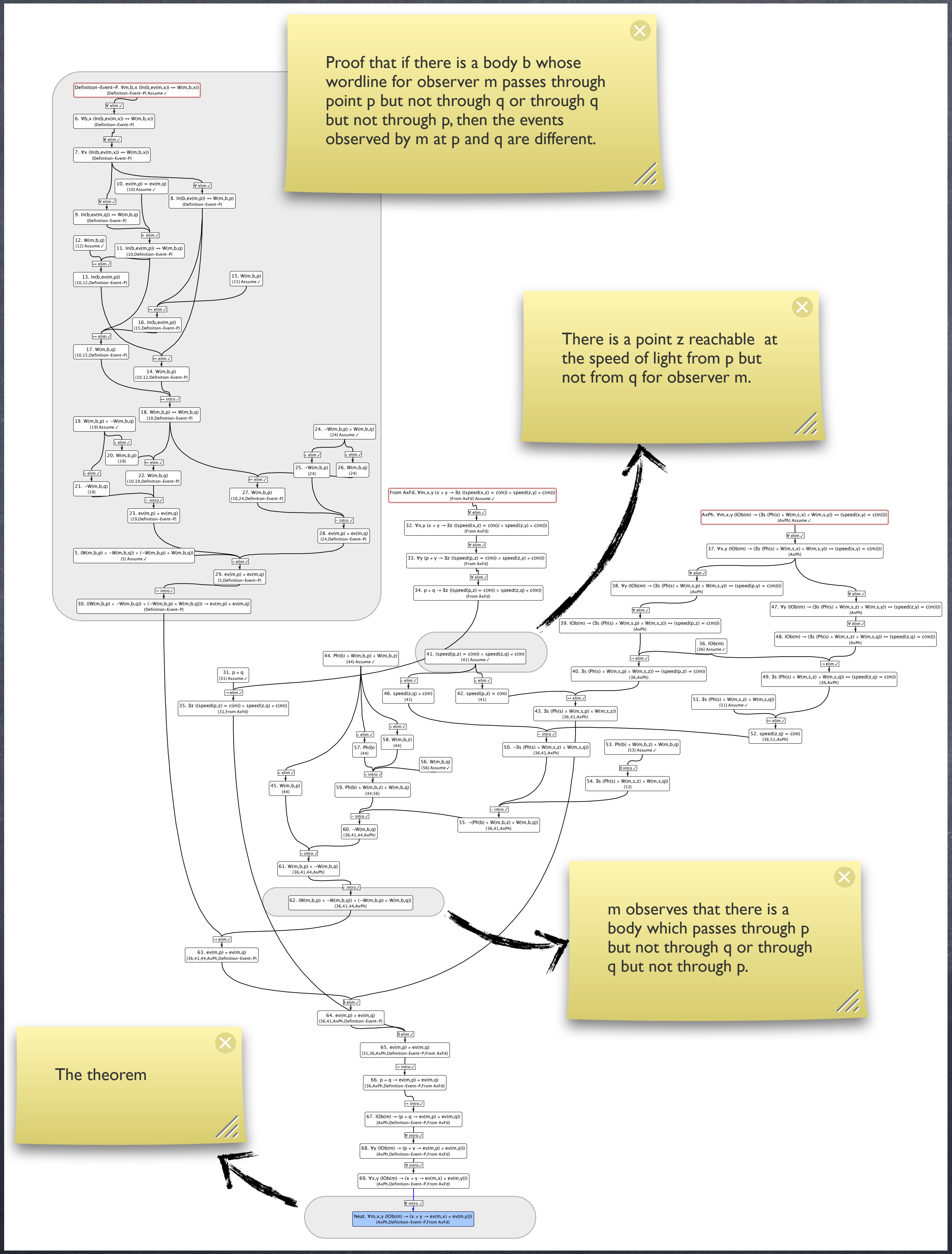
observe observes(ph,r) $\wedge$ through(r,l_1) $\wedge$ through(p,l_1) $\wedge$ $\neg$through (q,l_1)

$\Delta_2 \rightarrow \Delta_4$ by D;Δ with AxFd

Δ_5 by claim ph’ at r’

observe observes(ph’,r’) $\wedge$ through(r’,l_2) $\wedge$ through(p,l_2) $\wedge$ $\neg$through (q,l_2)

Verified


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

Demo!

Further (Special Theory) Proofs

- Time Dilation
- Length Contraction

Objection: How are our proofs different from purely sentential ones?

To Do: Proof discovery with Vivid-like ω-DPL

Köszönöm szépen