A Mechanized Investigation of an Axiomatic System for Minkowski Spacetime

Mathis Gerdes

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2021

Abstract

Einstein's theory of special relativity replaced the Euclidean space at the heart of Galilean physics with the new geometry of Minkowski space. Aiming to establishing a verified foundation for special relativity, this MSc project continues the mechanization of an axiomatic system for Minkowski space developed by Schutz in the interactive theorem prover Isabelle/HOL. First, the existing partial formalization is critically reviewed and several changes made to it are discussed. A new mechanization of the third theorem of collinearity introduced in Schutz's monograph is discussed. This required the development of new rigorous proofs capturing geometric intuitions which Schutz apparently derives from pictorial representations. Techniques to avoid combinatorial explosions arising in the mechanization of geometric proofs, by capturing without loss of generality notions, are discussed.

Acknowledgements

I would like to thank Jacques Fleuriot, Jake Palmer and Richard Schmoetten for their advice and assistance. Their feedback and insights were indispensable, and without them there would have been no project to continue.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Mathis Gerdes*)

Table of Contents

Chapter 1

Introduction

The predominant Galilean conception of space and time in physics was overturned in the beginning of the 20th century, when the empirical observation of a universally constant speed of light led Einstein to the theory of special relativity (SR) [1]. This was later developed into the theory of general relativity (GR), which incorporates the effect of gravity by treating it as the geometric curvature of a unified spacetime. Crucial for this development was the work of Minkowski, who turned Einstein's empirical description of SR into a systematic formalism in terms of the so called Minkowski spacetime[1], a flat non-Euclidean geometric space[2] [4].

Although Minkowski's spacetime formalism is now the standard description of relativity in physics, it was initially met with scepticism for being overly complex [5]. One reason for this is that Minkowski's linear-algebraic description of space and time is farther removed from our intuition than Einstein's initial, more empirical, description in terms of clocks, distances and movement through space. This suggests the problem of formalizing Minkowski spacetime as an axiomatic geometric space, using minimal building blocks such as events (points in space *and* time) and paths between events, analogous to the description of Euclidean geometry by a set of intuitively appealing and comprehensible axioms [6].

One such effort has been made by Schutz [7, 8, 9], which eventually led to a system of fifteen independent axioms that describe Minkowski spacetime. From these, the geometry is constructed in terms of derived theorems, and the standard linear-algebraic

[1] Minkowski spacetime may be referred to as both space and spacetime, one referring to it as a geometric space and the other as the unification of *physical* space and time.

[2] The metric of Minkowski space is flat. Gravity in GR manifests itself in local perturbations of the Minkowski metric, leading to curvature. A basic introduction to SR will be given in section 2.1. For an in-depth introduction to both SR and GR see any standard physics textbook on the subject [2, 3].

description of Minkowski space is eventually shown to be a model.

The aim of this project is to create a machine-checked formalization of Schutz's axioms and theorems as outlined in his 1997 monograph *Independent axioms for Minkowski space-time*[9] using the mechanized proof assistant Isabelle, continuing the work of two prior MSc projects with the same goal [10, 11, 12]. In these prior projects, mechanizations of the axioms described in Schutz's second chapter[3] as well as the theorems of chapter three (with the exception of a partial result of Theorem 12) were completed. The objective of this project is to advance the mechanization of Schutz's work and begin with the mechanization of chapter four. The mechanization of all three parts of Theorem 15 were completed, which involved filling in several lemmas that express geometric intuitions which Schutz appears to have taken for granted. Additionally, several existing mechanizations of definitions and derived objects were modified and corrected.

A brief introduction to special relativity, its axiomatization, and to Isabelle in the context of higher order logic (HOL) is given in section 2, as required to follow the main results. Since this project is, as stated, the direct continuation of two previous MSc projects, it was necessary to split it into two phases. In the first, the existing mechanized formalization in Isabelle was studied, which led to several improvements and corrections. These are explained and justified in section 3, together with a brief outline of Schutz's axiomatic system itself and relevant existing mechanizations.

The second phase of the project consists of the mechanization of Schutz's Theorem 15 and is discussed in section 4. Schutz makes considerable use of geometric intuition, which means that great care needed to be taken in finding appropriate definitions for several of his definitions. For the same reason, his proofs needed to be expanded by several lemmas he may have considered obvious as well as by a rigorous treatment of "without loss of generality" (WLOG) assumptions and case distinctions.

[3] If not otherwise stated, all following references to specific chapters, Schutz's axioms or his original formulation of theorems refer to his 1997 monograph [9].

Chapter 2

Background

A brief description of special relativity in terms of Minkowski's spacetime formalism is given in section 2.1. This aims to remind the reader of the general setting, but also serves as a demonstration of the formalism being rather far removed from intuitive concepts about space and time, which leads to the motivation for creating a (machine checked) geometric axiomatization of Minkowski space discussed in section 2.2. The interactive theorem prover Isabelle/HOL is introduced in section 2.3, which is necessary to understand the difficulties in moving from Schutz's pen-and-paper proofs to a mechanized version. Schutz's axiomatic system of Minkowski space is summarized in section 3, together with several relevant formalizations of definitions, axioms and theorems in Isabelle inherited from the previous MSc projects.

2.1 Special Relativity

Before Einstein, space and time were widely thought to adhere to Galilean relativity. An object moving at velocity v for observer O_1, for example, would move at velocity $v - v_0$ for an observer O_2 who moves at velocity v_0 with respect to O_2. These kind of linear transformations from one observer to the other have the characteristic property that they leave the Euclidean metric $dl^2 = dx^2 + dy^2 + dz^2$ invariant[1]. A moving observer measures the same spacial distances as an observer at rest. Motivated by this, physical space can be treated as a Euclidean space separate from time, with time parameterizing paths in this space.

If, in the above example, the object is a beam of light moving at the velocity c

[1]This notation for a metric is common in physics. It describes how an infinitesimal length dl is calculated in terms of displacements dx, dy, dz along each axis.

for O_1, we would expect O_2 to measure a speed of light of $c - v_0$. This was falsified experimentally, where the speed of light was always measured to be c for all observers, calling for a new theory. The defining property for this new theory is that all physical transformations[2] between observers must leave the speed of light $c^2 = (dx^2 + dy^2 + dz^2)/dt^2$ constant, that is we must always have $dx^2 + dy^2 + dz^2 - c^2dt^2 = 0$ when dx, dy, dz are distances light has travelled in a time dt along each axis. This eventually leads to a new geometry called Minkowski space [3], in which the following defining metric is left invariant by transformations between observers:

$$ds^2 = dx^2 + dy^2 + dz^2 - c^2dt^2\,, \tag{2.1}$$

With this theory, physical space can no longer be treated as a separate Euclidean space but is unified with time into a single geometric space. Points must now be given in three spacial and one temporal coordinate, and are thus typically referred to as events. Paths (in general not necessarily "straight") between two events are physical trajectories, but importantly not all events have paths connecting them. This is intuitively clear since it is not physical to move from one event to another event that has the same time coordinate. Mathematically, this manifests itself in these kind of events being separated by an imaginary distance:

$$ds = \sqrt{dx^2 + dy^2 + dz^2 - c^2dt^2} = \sqrt{-c^2dt^2}\,. \tag{2.2}$$

2.2 Formalization as an Axiomatic System

The comparison between the coordinate-based spacetime formalism for Minkowski space and its treatment in terms of axiomatic geometry is analogous to the different descriptions of Euclidean space. Euclidean space can either be defined in terms of linear algebra as a vector space $\mathbb{R}^n$, or it can be defined as an axiomatic system in the language of synthetic geometry, using undefined primitive objects and relations like points and collinearity, as famously done in Euclid's *Elements* [13].

While for Euclidean geometry the axiomatic treatment was developed before the modern linear algebra version, Minkowski space was first introduced in these terms. Axiomatic systems of Euclidean geometry have received considerable attention since Euclid, notably by Hilbert [14], who called for a general effort to axiomatize all areas of physics [15]. An early axiomatic description of Minkowski space was proposed by

[2] "Physical" in this context means "physically possible" such as changes in position and velocity.

Robb [16] and later further developed by Mundy [17], although compared to Euclidean geometry, relatively little attention has been paid to this effort.

Schutz's axiomatic system of Minkowski space was greatly influenced by those of Euclidean space. He specifically cites developments made by Hilbert [14], Veblen [18] and Moore [19], whose results are used in several of Schutz's theorems. Schutz's 1997 monograph, which is the main work considered for this project, builds on top of several previous approaches [20, 21], including his own earlier work [8]. An alternative axiomatic system using only first order logic (FOL) was proposed by Goldblatt [22, 23], which was criticized by Schutz for allowing non-isomorphic models[3]. In a more recent effort, a FOL formalization of special relativity was mechanized in Isabelle/HOL [24, 25]. It focuses more on physical predictions, however, and cannot be directly compared to the geometric axiomatization considered here [4].

2.3 Mechanized Formalization with Isabelle/HOL

Defining Minkowski space in terms of an axiomatic system gives it a philosophically appealing foundation. Constructing a theory based on a small set of comprehensible axioms using rigorous deduction increases the confidence in its results and internal consistency. As with Euclid's fifth postulate, it also facilitates the exploration of variant theories obtained by modifying or removing axioms.

All of the above benefits apply even more to a mechanized formalization of the axiomatic system. Using an interactive theorem prover, axioms, theorems and proofs can be expressed in a machine readable way such that proofs use only machine-verified logical steps. Assuming the soundness of the underlying software, this lends even greater confidence to the resulting theory than the corresponding pen-and-paper version. Translating an existing system of axioms, including theorems and proofs, into the framework of an interactive theorem prover thus serves to validate the line of reasoning and the logical consistency of the axioms. If the set of axioms are insufficient to prove the claimed theorems, this will become apparent by a failure to mechanize their proofs. Since traditional expositions of mathematical proofs often involve considerable intuition, "obvious" steps or minor errors, mechanizing a theory also serves as an opportunity to clarify proofs and make them more rigorous.

[3] This means more than one distinct vector space satisfies the given set of axioms. Schutz's model is categorical, which means all models satisfying his axioms are isomorphic (i.e. equivalent) to each other.

[4] It includes a theory about accelerated observers *inside of* Minkowski space, for example, while Schutz's axiomatic system describes the geometry *itself* without discussing physical implications.

The interactive theorem prover used for this project is Isabelle [26] within the higher order logic (HOL) framework. An introduction to Isabelle/HOL, its type system and its proof syntax is given below, as necessary to follow the subsequent discussion.

2.3.1 Formalizing Higher Order Logic

While Isabelle supports other underlying logic systems such as ZF set theory [27, 28], we will use the version which implements HOL [29]. HOL is defined by its ability to quantify over arbitrarily complex terms, not only over individual non-logical objects as in "$\forall n \in \mathbb{N}. n < n + 1$" (first order), or predicates as in "$\forall P.\ P\ x\ \rightarrow\ P\ x$" (second order). Isabelle is an LCF-style theorem prover [30, 31] and realizes HOL using a typed version of the λ-calculus [32], which will be explained in the next section.

2.3.2 Isabelle's Type System

Isabelle/HOL[33] is built on a type system similar to that of functional programming languages. A set of base types are provided, including booleans `bool` and natural numbers `nat`, as well as type constructors like `list` and `set`. One may also work with an abstract type using type variables, which are denoted with a prefixed apostrophe like `'a`. Valid instantiations of a type variable are constrained by postulating specific properties the type must satisfy, which will become clearer in section 2.3.5 on locales. The type of a function mapping two natural numbers to a third natural number is denoted as `nat ⇒ nat ⇒ nat`. An example of such a function, which can be expressed in Isabelle using the familiar λ-calculus, is `λ x y :: nat. Suc (x + y)`. Here, the notation `:: nat` indicates that both `x` and `y` have the type of a natural number. The function `Suc :: nat ⇒ nat` takes a natural number x to its successor $x+1$. Functions are always total over the declared type. It is not possible to declare a function to map from one specified set (of elements of some type) to another set.

2.3.3 HOL and Meta-Level Logic

Object level logic is expressed in Isabelle using the familiar quantifiers ∀ and ∃, and the arrows $\longrightarrow$ and $\longleftrightarrow$ denote implication and equivalence. Since we use higher order logic, this object-level syntax is also referred to as HOL syntax.

In order to facilitate logical deduction, Isabelle has two levels of syntax in which logic is expressed. Deduction is done via the application of truth preserving rules.

Consider, for example, that the goal is to prove some statement `A₁ x` about the object `x`, and there is a lemma which states `B₁ ?a` and `B₂ ?a` together imply `A₁ ?a`. The question mark in front of `?a` is indicates that the statement is true for all instantiations of the lemma, with `?a` replaced by an expression of the correct type. Isabelle can then apply this lemma backwards to replace the goal `A₁ x` with the subgoals `B₁ x` and `B₂ x`. This kind of instantiation-ready quantification of a variable is expressed in Isabelle with the symbol $\bigwedge$, and implication is written as $\Longrightarrow$. This so-called meta-level syntax is logically equivalent to the corresponding HOL syntax but treated slightly differently by Isabelle. The above lemma is then stated in Isabelle as $\bigwedge$`a. B₁ a` $\Longrightarrow$ `B₂ a` $\Longrightarrow$ `A₁ a`, or in the equivalent compressed version $\bigwedge$`a. ⟦B₁ a; B₂ a⟧` $\Longrightarrow$ `A₁ a`.

2.3.4 Isar Language and Proofs

The primary features of interactive theorem provers are the machine-verification of proofs and advanced algorithms which can prove many steps automatically. While one may thus trust a machine-checked theorem without verifying it by hand (assuming one trusts the underlying logic kernel), it is still important for proofs to be comprehensible to humans. There is no algorithm which can decide which theories are worthwhile to develop and which theorems are important to prove. It is thus still important for humans to develop an intuition, which is built up by deriving results from the postulated axioms. Furthermore, while Isabelle provides the *sledgehammer* tool [34] to search for a proof of a given (sub-)goal, more complex proofs must still be developed manually.

Isabelle provides the Isar language [35], which aims to facilitate both human and machine readability. It combines the above backward reasoning, well-suited for the automated solvers, with the forward reasoning style common in mathematics. Since Isar is designed to be intuitive to understand, most of the results here can be followed with a basic understanding of it. Consider, for example, the following lemma which shows that `[a;b;c]` implies `a≠b`. The relation `[a;b;c]` will be introduced later.

```
1  lemma abc_ab_neq:
2    fixes a b c :: 'a
3    assumes abc: "[a;b;c]"
4    shows "a ≠ b"
5  proof (rule notI)
6    assume "a = b"
7    hence "[c;a;a]" using abc abc_sym by simp
8    hence  "[c;a;c]" using abc abc_bcd_abd ⟨a=b⟩ by blast
9    then show "False" using abc_ac_neq by blast
10 qed
```

The lemma is introduced with the keyword **lemma** followed by a name which can be used to refer to its results in later proofs. The keyword **theorem** is equivalent and indicates a higher level of importance. In the second line, we introduce three variables `a`, `b` and `c`, and declare them to be of type `'a`. Using the keyword **assumes**, the assumptions of the lemma are sated. Here, the assumption is given the label `abc`, by which we can refer to it later. This is especially helpful if there are multiple assumptions, or if we want to refer to intermediate results in a more complex proof. The statement of the lemma is completed with the **shows** keyword, followed by whatever the lemma concludes. Lines two and three could be expressed in a single line using Isabelle's meta-level syntax: "⋀ a b c :: 'a. [a;b;c] ⟹ a ≠ b".

The proof of the lemma is begun with the **proof** keyword, optionally followed by a proof method which modifies the goal of the following proof. Important examples are proof by contradiction, which replaces the goal `A` by the goal ¬A ⟹ False, case splitting, and induction. Here, the proof method is an application of the following rule: notI: "(P⟹False) ⟹ ¬P". Isabelle automatically tries to apply a proof method if none is given, which can be prevented by writing **proof** -.

Above, the proof opens by stating the assumption which matches the proof's goal. This is generally followed by a list of derived statements, culminating in a final statement which matches the conclusion of the proof's goal and is preceded by the keyword **show**. There are several abbreviations which make Isar proofs more readable. A general statement claimed to be true is introduced with the keyword **have**. If the proof makes use of the preceding statement, this is indicated by **then have**, which is abbreviated to **hence**. A shorthand for **then show** is **thus**. Other such keywords used here should become clear from their context.

The proof of a statement itself consists of two components. Besides the keyword **then**, further facts can be made available by listing their names after the **using** keyword. The facts used in the above proofs are axioms listed in the example for locales in the next section. Finally, the method used to derive the statement given the listed facts is chosen with the **by** keyword. Available proof methods include `simp` and `auto` and automatic theorem provers like `blast` and `metis`. More difficult statements may be proved using their own **proof** ... **qed** block. In this text, a proof method following the **by** keyword is always meant to indicate that the stated proof method successfully concludes without error. For brevity, a proof may be represented by **<proof>**, and omitted parts are indicated by ... .

A successful proof is terminated with **qed**. An unsuccessful proof can be termi-

nated using two keywords: **oops** terminates a proof but does not allow the lemma or statement to be used in further deductions, while **sorry** allows this. The latter should be used carefully, as a contradiction may be introduced in this way and may make future results meaningless (anything can be validly deduced from `False`).

2.3.5 Locales

Theorems and lemmas can be organized into locales, which make it possible to safely introduce a set of axioms assumed to be true. Locales can be built on top of each other, inheriting all contained facts. This allows a theory to be built up systematically, with axioms and results grouped together thematically. In this way, the influence of axioms can be made explicit, and sub-theories can be studied independently. Furthermore, axioms are only true inside the locale so the effect of faulty axioms is limited.

To understand how locales work, consider the following example which defines the `Betweenness` locale discussed in section 3.3. It introduces the axioms that were used in the proof of `abc_ab_neq` above.

```
1 locale Betweenness =
2   fixes between :: "'a ⇒ 'a ⇒ 'a ⇒ bool" ("[_;_;_]")
3   assumes
4     (* O2 *) abc_sym: "[a;b;c] ⟹ [c;b;a]" and
5     (* O3 *) abc_ac_neq: "[a;b;c] ⟹ a ≠ c" and
6     (* O4 *) abc_bcd_abd: "⟦[a;b;c]; [b;c;d]⟧ ⟹ [a;b;d]"
7 begin (* lemma abc_ab_neq: ... *) end
```

Just like a lemma, the locale begins with a block containing **fixes** and **assumes**. In the second line, we fix a function named `between` with the type `'a ⇒ 'a ⇒ 'a ⇒ bool`. Implicitly, this also introduces the type variable `'a` over which the relation is defined. A nice feature of Isabelle is that we can introduce a notation for our function. In this case, the notation `[a;b;c]` is defined to be equivalent to the predicate `between a b c`. At this point, we have fixed some undefined ternary relation and introduced a notation for it. To be able to reason about it, we introduce three axioms following the **assumes** keyword which specify its properties. All fixed variables and assumptions are available within the body of the locale, which is delimited by **begin ... end**. Locales can be built on top of other locales, in which case they inherit all axioms and definitions as well as proven theorems and lemmas.

Chapter 3

Refactoring the Formalization of Schutz's Axiomatic System

Schutz introduces Minkowski space $\mathcal{M}$ in terms of undefined primitives:

$$\mathcal{M} = \langle \mathcal{E}, \mathcal{P}, [\cdot\ \cdot\ \cdot] \rangle .$$

Here, $\mathcal{E}$ is some set of elements called events. $\mathcal{P}$ is restricted to contain subsets of $\mathcal{E}$. Elements of $\mathcal{P}$ are called paths. $[\cdot\ \cdot\ \cdot]$ is a ternary relation of events called betweenness, which is further discussed in section 3.3.

In total, Schutz introduces fifteen axioms to describe Minkowski space, grouped into six axioms of order, seven axioms of incidence, the Axiom Of symmetry or isotropy, and the Axiom Of continuity. All of these axioms have been fully formalized in Isabelle in the previous two MSc projects [10, 12] and, apart from some small modifications discussed below, retained for this project.

To be able to continue the formalization of Schutz's work, the first part of the project consisted of critically studying the existing one. While the prior mechanization was overall found to be of very high quality and well suited to expand upon, several improvements and small corrections were made. The following sections summarize the most important changes made. Where necessary, Schutz's formulation and the mechanized formalization as completed in the previous MSc projects are briefly introduced. A full discussion of the differences between Schutz's prose and their existing mechanizations is not repeated here, and can be found elsewhere [10, 11, 12].

3.1 Events and Paths

We first introduce some basic notions about paths and events. Schutz's space fundamentally consists of a set $\mathcal{E}$ of elements called events. Some sets of events are called paths, and these make up the set $\mathcal{P}$. In Isabelle, we introduce the primitives of Schutz's Minkowski space inside a locale `MinkowskiPrimitive`:

```
locale MinkowskiPrimitive =
  fixes ℰ  :: "'a set"
    and 𝒫  :: "('a set) set"
  assumes paths_in_pow_events: "𝒫 ⊆ Pow ℰ"
      and ... (* Axioms I1–I3 shown below *)
```

Here, we introduced a type variable `'a` for events. In order to show that for example $\mathbb{R}^4$ is a model for Schutz's axioms, we would instantiate the `'a` with a type for that vector space. Inside the axiomatic system, the type is kept unspecified.

We will see below that a path is uniquely defined by two distinct elements that lie on it. Schutz thus often denotes a path with two such elements a and b simply as ab. To capture this, we introduce the predicate `path`:

```
abbreviation path :: "'a ⇒ 'a ⇒ 'a set ⇒ bool" where
  "path a b ab ≡ ab∈𝒫 ∧ a∈ab ∧ b∈ab ∧ a≠b"
```

In order to state that two events lie on a path without specifying it, we introduce:

```
abbreviation path_ex :: "'a ⇒ 'a ⇒ bool" where
  "path_ex a b ≡ ∃ab. path a b ab"
```

3.2 Notation

In the previous MSc projects, Schutz's betweenness relation $[abc]$ was denoted as `[[ a b c]]`. The reason for this is that Schutz's notation would clash with the notation for lists with a single element and lead to ambiguous parse trees (the parser may think `a` is a function taking two arguments). While the parse ambiguity was successfully avoided by using double brackets, the notation still clashes with that of lists. For example, the list of a list of two numbers `[[1::nat,2::nat]]` fails to be parsed in the locale in which the double-bracket betwenness relation is defined. This does not technically pose a problem, as long as lists are not used for anything relating to the present formalization, and the list syntax can be disabled completely. The clash of notation is, nonetheless, stylistically unpleasing and would become a problem if we decided to use lists for some future problem.

A new notation has been introduced for this project, which aims to resolve any ambiguity with lists. Instead of Schutz's spaces between events in $[abc]$, we now use semicolons and retain the single bracket: `[a;b;c]`. This has an additional advantage over the original mechanized version. In more complex expressions, delimitation by semicolons (as opposed to spaces) alleviates the need for additional brackets. Instead of an expression like

```
[[(f (i-1)) (f i) (f (i+1))]],
```

we can now write the arguably easier to read version

```
[f (i-1); f i; f (i+1)].
```

Furthermore, Schutz later extends the betweenness ordering to four events. Using semicolons as delimitation, the expression `[a;b;c;d]` can be parsed without a problem, while the old notation led to parsing ambiguities. Again, `[[a b c d]]` may actually be the ternary ordering if `c` is a function taking a single argument. This ambiguity is avoided using the new notation.

Lastly, Schutz introduces the non-strict orderings $[abc]\!]$ and $[\![abc]\!]$ (further discussed in section 3.3.2). In the old mechanized notation, double brackets were already used for the ternary version, which led to the usage of a special bracket: ⟦`a b c`⟧. Unfortunately, this clashes with the meta-level syntax for assumptions in a lemma. Using the new notation, double brackets are freed up and we can define non-strict orderings as `[a b c]]` and `[[a b c]]`.

We will see below that, using Schutz's axioms, betweenness $[abc]$ implies the existence of a path that contains the events a, b and c, these events are distinct, and the betweenness relation is symmetric. This motivates the notation `[a;b‖Q]` for `path a b Q` and `[a;b]` for `path_ex a b`. Even though ternary ordering is not derived from the binary relation `path_ex`, a unified notation invokes the correct intuition.

Based on the above changes to betweenness, the notation for chains (see section 3.5) was also modified to be consistent. A summary of all changes to the notation in Isabelle can be found in table B.1 in the appendix.

3.3 Axioms of Order and the Betweenness Locale

The relation Schutz uses to give a geometric structure to the set of events and paths is the ternary betweenness relation $[\cdot\ \cdot\ \cdot]$. Paths in Schutz's system correspond to

the physical paths of inertial particles. Intuitively, the betweeness relation $[abc]$ can be understood as the statement that a, b and c are events which occur on a path and are ordered temporally[1], i.e. the event b happens between the events a and c. The properties of the betweenness relation are defined by the six axioms of order[2] O1 to O6 shown in table 3.1 below. In the prior mechanization, Axioms O1-O5 were introduced

Table 3.1: Mechanized axioms of order O1-O5. The notation `dist3 a b c` is an abbreviation for a,b and c being distinct. Axiom O6 omitted for brevity.

Axiom (Name)	Statement
O1 (abc_ex_path)	`[a;b;c]` $\Longrightarrow$ $\exists$`Q`$\in\mathcal{P}$. `a,b,c` $\in$ `Q`
O2 (abc_sym)	`[a;b;c]` $\Longrightarrow$ `[c;b;a]`
O3 (abc_ac_neq)	`[a;b;c]` $\Longrightarrow$ `a` $\neq$ `c`
O4 (abc_bcd_abd)	$[\![$`[a;b;c]; [b;c;d]`$]\!]$ $\Longrightarrow$ `[a;b;d]`
O5 (some_betw)	$[\![$`Q`$\in\mathcal{P}$`; a,b,c`$\in$`Q; dist3 a b c`$]\!]$ $\Longrightarrow$ `[a;b;c]` $\vee$ `[b;c;a]` $\vee$ `[c;a;b]`

all together in the locale `MinkowksiBetweenness`, which is built directly on top of the lowest-level locale `MinkowskiPrimitive`. During this project, it was noticed that axioms O2-O4 can be stated independently of the sets $\mathcal{E}$ and $\mathcal{P}$ and define a kind of standalone "theory of betweenness". For this reason, these axioms were moved into their own `Betweenness` locale. Several meaningful lemmas can be proved within this locale alone, justifying the separation. It may be possible to move further results into the locale (e.g. results about chains), stressing their dependence on only a small number of assumptions. This was not further pursued. An outline of the new locale was shown in section 2.3.5. Here, we list several conceptually important results that can be proven inside of it:

- Events satisfying betweenness are distinct. For this we use the abbreviation `dist3 a b c` $\equiv$ `a`$\neq$`b` $\wedge$ `b`$\neq$`c` $\wedge$ `c`$\neq$`a`:

```
lemma abc_abc_neq: "[a;b;c] ⟹ dist3 a b c"
```

- Alternate version of transitivity property asserted by Axiom O4:

```
lemma abc_bcd_acd: "⟦[a;b;c]; [b;c;d]⟧ ⟹ [a;c;d]"
```

[1] Schutz's system does not pick a positive time direction since the betweenness relation is symmetric, which means that we cannot say whether the events are ordered or ordered in reverse.

[2] The statement here does not follow Isabelle's syntax strictly in order to improve legibility.

- Given `[a;b;c]`, all other permutations (except `[c;b;a]`) are false. Since betweenness is symmetric (Axiom O2) there are only four distinct permutations:

```
lemma abc_only_cba:
  assumes "[a;b;c]"
  shows "¬[b;a;c] ∧ ¬[a;c;b] ∧ ¬[b;c;a] ∧ ¬[c;a;b]"
```

3.3.1 Kinematic Triangle

Based on the properties introduced above, we can define a geometric constellation called a kinematic triangle. This notion plays a central role in Theorem 15. In this context, we furthermore note a modification that was made to its definition in the existing formalization.

Schutz calls three distinct events $\{a,b,c\}$ a kinematic triangle if each pair of events belongs to one of three distinct paths. This corresponds exactly to the usual concept of a triangle specified by its vertices. A triangle is denoted `△ a b c`, both by Schutz and in Isabelle. The existing mechanization of the kinematic triangle was as follows:

```
1 definition kinematic_triangle ::
2   "'a ⇒ 'a ⇒ 'a ⇒ bool"
3 where "△ a b c ≡
4   a ∈ ℰ ∧ b ∈ ℰ ∧ c ∈ ℰ ∧ a ≠ b ∧ a ≠ c ∧ b ≠ c
5   ∧ (∃Q∈𝒫. ∃R∈𝒫. Q ≠ R ∧ (∃S∈𝒫. Q ≠ S ∧ R ≠ S
6         ∧ a ∈ Q ∧ b ∈ Q ∧ a ∈ R ∧ c ∈ R ∧ b ∈ S ∧ c ∈ S))
```

Using the new notation for a path crossing through two points, we can introduce a much shorter version which is easier compare with Schutz's original formulation:

```
1 definition kinematic_triangle :: "'a ⇒ 'a ⇒ 'a ⇒ bool"
2 where "△ a b c ≡
3   ∃Q∈𝒫. ∃R∈𝒫. ∃S∈𝒫.
4     dist3 Q R S ∧ [a;b‖Q] ∧ [a;c‖R] ∧ [b;c‖S]"
```

The two definitions were shown to be equivalent using a mechanized proof.

3.3.2 Non-Strict Orderings

A more meaningful change was made to the definition of non-strict orderings. Schutz literally defines these as[3] $[ab\ldots ef\rrbracket := [ab\ldots ef]$ or $e = f$ which (for the case of four elements) was mechanized previously as

[3] Orderings of four or more events denote chains, which are discussed in section 3.5.2. Specifically, $[abcd] = [abc] \wedge [bcd]$.

```
abbreviation nonstrict_betw_right4 ::
  "'a ⇒ 'a ⇒ 'a ⇒ 'a ⇒ bool" where
  "nonstrict_betw_right a b c d ≡ [a;b;c;d] ∨ c=d"
```

With this definition, $[ab\ldots cc]$ would always be true, even if a and b are not connected by a path. While this interpretation of Schutz's text seems odd on its own, strong evidence that Schutz had another definition in mind comes from a reading of Lemma 2 for Theorem 15. In it appears an expression like $[abcd]\!]$. With the literal interpretation above, the event b would be completely unconstrained[4] if $c = d$. Furthermore, Schutz makes use of the fact that $[abc]$ without comment or case distinction. This suggests the following definition instead:

$$[abcd]\!] := [abcd] \ \vee \ (c = d \wedge [abc])\,. \tag{3.1}$$

Therefore, the mechanized version has been replaced by the following:

```
abbreviation nonstrict_betw_right4 ::
  "'a ⇒ 'a ⇒ 'a ⇒ 'a ⇒ bool" where
  "nonstrict_betw_right a b c d ≡
        [a;b;c;d] ∨ ([a;b;c] ∧ c=d)"
```

All existing mechanized proofs were successfully modified to use this new definition.

3.4 Axioms of Incidence

Besides the axioms of order, several important properties of Schutz's system which are used in the following are introduced through the axioms of incidence. Table 3.2 shows these axioms as formulated in the existing mechanization. Axiom I3 states

Table 3.2: Mechanized axioms of incidence I1-I3. Axioms I4-I7 omitted for brevity.

Axiom (Name)	Statement
I1 (nonempty_events)	$\mathcal{P} \neq \{\}$
I2 (events_paths)	$[\![a,b \in \mathcal{E};\ a \neq b]\!]$ $\Longrightarrow \exists R,S \in \mathcal{P}.\ a \in R \wedge b \in S \wedge R \cap S \neq \{\}$
I3 (eq_paths)	$[\![R,S \in \mathcal{P};\ a,b \in R;\ a,b \in S;\ a \neq b]\!] \Longrightarrow R = S$

the property that a path is fully specified in terms of two distinct events that lie on it. An important implication is that two events which both lie in the intersection of two

[4] Apart from one other assumption which is not sufficient to prove the lemma.

distinct paths must be the same. This corresponds to our geometric intuition that two distinct straight lines meet exactly once, which Schutz often uses implicitly.

3.5 Consistent Definition of Chains

Schutz defines chains as a sequence of events on a path Q, denoted either as $[Q_0\ Q_1\ \cdots]$ if infinite or $[Q_0\ Q_1\ \cdots\ Q_{n-1}]$ if finite , with the following properties:

- Sequence has two elements: The two elements are distinct.
- More than two elements: For all $i \geq 2$, have $[Q_{i-2}\ Q_{i-1}\ Q_i]$.[5]

Since in the second property betweenness ordering is only demanded for adjacent elements of the chain, we say the chain is locally ordered. Axiom O4 (see table 3.1) postulates a kind of transitivity property for chains, however, which leads to the chains being totally ordered: For all $i < j < k$, have $[Q_i\ Q_j\ Q_k]$.

In fact, a proof that local ordering implies total ordering given Axioms O4 and O2 is the content of Schutz's Theorem 2. Schutz explicitly states this result only for finite chains. However, the result immediately implies total ordering for infinite chains also: For three fixed but arbitrary elements, we can restrict the infinite chain to the finite sub-chain terminating at the largest relevant index and obtain the desired total ordering via Theorem 2. Besides total ordering, chains gain a second property using the axioms. Using Axiom O3 and total ordering, it can be shown that all elements of a chain are distinct.

The definition of chains in Isabelle was a major topic in both prior MSc projects. There is no primitive in Isabelle that corresponds directly to the mathematical notion of sequences that Schutz uses. While lists could be used for finite chains, infinite lists are not defined and lists are always constructed by recursive insertion which may complicate proofs and definitions. The solution to this, which was already introduced in the first MSc project, is to replace the sequence with a set X containing its elements and an index function:

$$f : \{0, \ldots, |X| - 1\} \longrightarrow X\,. \tag{3.2}$$

The sequence $[a\ b]$, for example, would be represented by $f = \{0 \mapsto a, 1 \mapsto b\}$ and the set $X = \{a, b\}$. Since we want f to be an index function for all elements in X, we

[5] Necessary bounds on indices, such as $i < n$ for a finite chain, are always implied.

require f to be surjective. In general, f need not be injective. For Schutz's chains this follows, however, from the fact that all elements of a chain are distinct, making f bijective.

We see from the above discussion that Schutz's definition of chains can be rapidly enhanced. Using the axioms of order, local ordering can be elevated to total ordering and we can prove that all chain elements are distinct. Since we do not have the goal to mechanize Schutz's formulations as literally as possible, we are at liberty to immediately incorporate further facts into the definition of chains, especially if this makes our work simpler. This is, in fact, exactly what was done in the previous MSc projects.

In the first project [10], chains were defined based on a total ordering of chains, making use of both the fact that chains are totally ordered and that elements are distinct. This effectively made Schutz's Theorem 2 obsolete, as it was immediately implied by the definition. During the second project [12], another kind of ordering, termed local ordering, was introduced to define an additional notion of chains, which could be used to stay closer to Schutz's formulations of proofs. It did not, however, exactly match Schutz's definition either, as it still uses the fact that elements of chains are distinct.

Unfortunately, the past struggle with the definition of orderings and chains has led to a proliferation of disparate versions in the formalization inherited from the prior projects, making these parts difficult to follow. This motivated an effort, undertaken in the present project, to rigorously compare the different definitions of orderings with Schutz's definition, and to refactor the mechanized formalization to use consistent definitions of orderings and chains.

In the next section, we will consider three possible definitions of orderings. All aim to formalize sequences, which are either locally or totally ordered, using a set and an index function. These form the basis for the definition of chains, which is discussed in section 3.5.2. The second and third orderings are exactly the local and total ordering introduced in the previous projects. The first ordering, termed weak ordering, is new and aims to replicate Schutz's definition faithfully.

3.5.1 Weak, Local and Total Ordering

The different kind of ordered sequences are defined here for a general ternary ordering `ord`. The definitions in this section are written in a verbose style which aims to closely resemble their mechanized versions in Isabelle while being considerably more readable (several details not important to the present discussion are ignored and some liberties

were taken with syntax). The actual Isabelle versions of these definitions are discussed in section B.3 of the appendix. For the version reproducing Schutz's definition most literally, the elements are not assumed to be distinct and only local ordering is asserted:

Definition 1 (`weak_ordering f ord X N`)
A weakly ordered sequence of `N` elements is defined by the set `X :: 'a` with index function `f :: nat ⇒ 'a` if the function is surjective (`∀x∈X. ∃i<N. f i = x`) and it defines a length `N` sequence (`∀i<N. f i ∈ X`) which is locally ordered:

```
∀i<N+2. ord (f i) (f (i+1)) (f (i+2)).
```

We allow $N \in \mathbb{N} \cup \{\infty\}$. How this is done is discussed in section B.3.1 of the appendix.

We can come to a slightly more convenient definition of ordering by realizing that chains always contain distinct events. Thus, we always have $|X| = N$ and we can get rid of the additional length-parameter:

Definition 2 (`local_ordering f ord X`)
A locally ordered sequence of `N` elements is defined by the set `X :: 'a` with index function `f :: nat ⇒ 'a` if the function is surjective (`∀x∈X. ∃i<N. f i = x`), and the sequence is a reordering of `X` (`∀n<|X|. f n ∈ X`) which is locally ordered:

```
∀i<|X|+2. ord (f i) (f (i+1)) (f (i+2)).
```

The first property states that `f` restricted to `{1,...,|X|}` is surjective onto `X` while second property asserts that `f` maps `{1,...,|X|}` into X. The third property is the same as for the weak ordering. If `X` is infinite, properties one and two simply state `f` corresponds to a surjective function $f : \mathbb{N} \longrightarrow X$, which makes it equivalent to weak ordering. For a finite set `X`, however, properties one and two imply that `f` restricted to `{1,...,|X|}` is a bijective function[6] $f : \{1,\ldots,|X|\} \longrightarrow X$, making the definition strictly stronger than the weak ordering. Finally, we can replace local ordering with total ordering:

Definition 3 (`total_ordering f ord X`)
A totally ordered sequence of `N` elements is defined by the set `X :: 'a` with index function `f :: nat ⇒ 'a` if the function is surjective (`∀x∈X. ∃i<N. f i = x`), and the sequence is a reordering of `X` (`∀n<|X|. f n ∈ X`) which is totally ordered:

[6]To see why, note that the second property implies the restricted function f maps between sets of equal cardinality. A surjective function between finite sets of equal cardinality is always injective.

```
∀i<j<k<|X|. ord (f i) (f j) (f k).
```

The definition of sequences for infinite sets `X` is equivalent in all three definitions. Without using the axioms of order, all allow elements of `X` to be repeated in the sequence. For finite sets, local and total ordering demand the index function to be bijective, which effects that each element of `X` appears exactly once in the sequence. For Schutz's betweenness relation, and using the axioms of order, all three definitions are equivalent.

3.5.2 Definitions of Chains

Since switching to weak ordering would necessitate the introduction of an additional variable for the chain length, it was decided to use local ordering for all definitions of chains. This is a compromise between complexity and proximity to Schutz's version. Additionally, proofs from the prior projects only needed to be modified slightly to prove equivalence between local orderings and total orderings given Axioms O2 and O4. A mechanized proof that weak ordering is equivalent to local and total ordering would have been time and has not been finished.

All of the mechanized formalization was refactored to use a consistent set of chain definitions. Appropriate equivalences with all old definitions were also shown in mechanized proofs. An elaborate explanation of all different old chain definitions and equivalence proofs would not be beneficial at this point. A short outline of the main new chain definitions is given in section B.3.4 of the appendix.

3.6 Summary

The above sections outline the most important changes made to the existing mechanization of Schutz's axiomatic system. A new locale describing the betweenness relation was introduced, in line with the goal of using distinct locales for logically independent sub-theories. A new notation solves several syntactic issues and makes the Isabelle code easier to read. The definition of kinematic triangles was simplified to align closely with Schutz's version. The various inconsistent existing definitions of chains were carefully studied and replaced by a single consistent kind. The existing proofs were refactored, leading to simplifications in several places. With the study and refactoring of the existing mechanization completed, the next section outlines the advances that were made in mechanizing further theorems introduced by Schutz.

Chapter 4

Third Collinearity Theorem

The third collinearity theorem (Theorem 15) establishes a correspondence between the arrangement of intersections of two paths with a kinematic triangle and whether they meet at an internal event. For easy reference, Schutz's statement of the theorem and his proofs in prose are given in section A of the appendix. The theorem proceeds in three parts. Part (i) of Theorem 15 was most straightforward to mechanize and does not involve conceptually new steps. It proves a partial result which is used in the other two parts of the theorem. Part (ii) was harder to mechanize and required careful considerations about how to formalize a notion Schutz describes in prose and on which the proof depends. While part (iii) is proved by Schutz in only two lines, claiming it to be an immediate consequence of previous results, it proved the hardest to formalize. Schutz seems to have used several non-trivial facts which may appear obvious when looking at pictorial representations but which are in fact non-trivial when proved rigorously using the axiomatic system.

In the next section, several common concepts and previous theorems are briefly reviewed. Afterwards, each part of the proof is dedicated its own section. This clarifies what work was necessary for each part of the theorem, and how much more complicated the mechanization is compared to Schutz's statement.

4.1 Relevant Definitions and Results

In this section, relevant definitions and theorems mechanized previously are briefly introduced. These include the notions of segments and intervals, the second collinearity theorem (Theorem 7) and Theorem 8. Schutz also uses Theorem 14, but his use was straightforward to mechanize and will thus not be discussed at length. Lastly, new

mechanizations for boundary and internal events of a kinematic triangle are discussed.

4.1.1 Segments and Intervals

Conceptually, segments and intervals correspond to the familiar open and a closed interval over real numbers. The segment between a and b, denoted as (ab), is the set of all elements x such that $[axb]$:

```
lemma segment_betw: "x ∈ segment a b ⟷ [a;x;b]"
```

The interval from a to b is the segment (ab) joint with the endpoints:

```
definition interval :: "'a ⇒ 'a ⇒ 'a set" where
  "interval a b ≡ insert b (insert a (segment a b))"
```

Both intervals and segments are manifestly symmetric.

4.1.2 Theorems 7 and 8

Theorem 7 states that a path which meets the extension of one side of a kinematic triangle and the inside of the nearest other side of the triangle also meets the inside of the last side of the triangle. Statements like that are often difficult to follow in prose form, and can be seen more easily from visual representations. For Theorem 7, this can be found in figure 4.1. The theorem was mechanized previously as

```
theorem (*7*) collinearity2:
  assumes "△ a b c"
      and "[b;c;d]" and "[c;e;a]" and "[d;e‖de]"
  shows "∃f∈de. [a;f;b] ∧ [d;e;f]"
```

Theorem 8 states there is no path as in figure 4.2 which touches each side of a triangle:

```
theorem (*8*) tri_betw_no_path:
  assumes "△ a b c"
      and "[a;b';c]" and "[b;c';a]" and "[c;a';b]"
  shows "¬(∃Q∈𝒫. a'∈Q ∧ b'∈Q ∧ c'∈Q)"
```

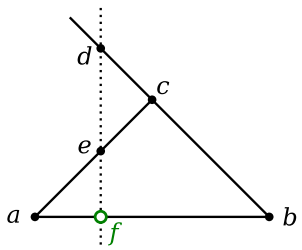

Figure 4.1: Visualization of Theorem 7.

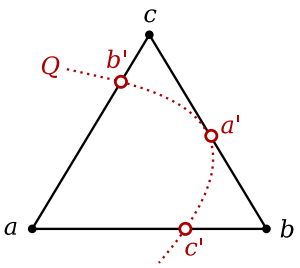

Figure 4.2: Visualization of Theorem 8.

4.1.3 Boundary and Internal Events

Schutz introduces the boundary and internal events as follows:

> Given a kinematic triangle $\triangle a_1a_2a_3$, the three intervals $|a_1a_2|$, $|a_2a_3|$, $|a_3a_1|$ are called *sides* and their union is called the *boundary* $\mathcal{B}(a_1,a_2,a_3)$. Any event between two events of distinct sides of the triangle is called an *internal event*. (Schutz [9, p. 31])

We first introduce the set of sides, which will become useful later:

```
abbreviation tri_sides :: "'a ⇒ 'a ⇒ 'a ⇒ 'a set set"
  where "tri_sides a b c ≡
    {interval a b, interval b c, interval c a}"
```

Note that the corner events of a triangle are distinct and, since intervals contain both endpoints, the above set thus has three distinct elements. The boundary is now simply:

```
abbreviation tri_boundary :: "'a ⇒ 'a ⇒ 'a ⇒ 'a set"
  where "ℬ{a b c} ≡ ⋃(tri_sides a b c)"
```

Imitating Schutz's notation, we denote the boundary set as `ℬ{a b c}`.

For the internal event, we first note that Schutz's language is slightly ambiguous. We could argue that two corners b, c of a triangle abc are on two distinct sides ab, bc, and thus every event in the segment (bc) must be considered internal. A reading of Theorem 15 (see section 4.2) strongly suggests, however, that this is not what Schutz had in mind. Instead, we assert the two events must not be in the same interval.

In formulating this definition, we encounter an issue which will reoccur frequently below. Namely, the triangle is symmetric under permutations of the corners, and we would like to simply draw two pictures as in figure 4.3 and claim these cover all definitions of internal events. We use the following abbreviation:

```
abbreviation permut3 :: ... where
  (* Any permutation *)
  "permut3 a b c a' b' c' ≡
    (a'=a ∧ b'=b ∧ c'=c) ∨ (a'=a ∧ b'=c ∧ c'=b) ∨ ..."
```

The event is an internal event, if for two points x, z there is a permutation of the triangle corners such that either of the two pictures in figure 4.3 is true:

```
definition internal_event :: " ... ⇒ bool" where
  "internal_event a b c e ≡
    (∃a' b' c' x z. permut3 a b c a' b' c' ∧
        [x;e;z] ∧ [a';x;b'] ∧ ([b';z;c'] ∨ z = c'))"
```

While an explicit version of this, which lists all possibilities, has been proven to be equivalent, it did not turn out to be more useful in the following proofs. In order to have a consistent notation with the boundary set, a new notation was introduced:

```
e ∈ ℐ{a b c} ⟷ internal_event a b c e.
```

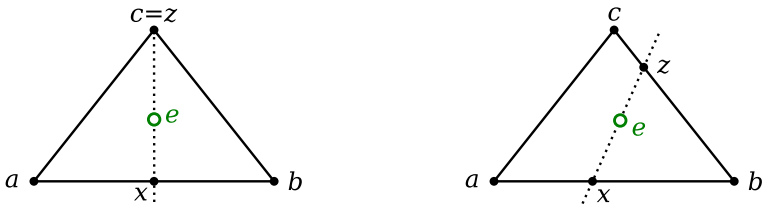

Figure 4.3: The two possible configurations in the definition of internal events.

Our reformulation of Schutz's definition could be stated as follows. An event is an internal event if it lies between two points x, z of the triangle's boundary which do not lie on the same interval. This is encoded by the following, conceptually pleasing but less useful definition, which was proved to be equivalent:

```
lemma internal_event_alt:
"internal_event a b c e ⟷ ∃x∈ℬ{a b c}. ∃z∈ℬ{a b c}.
  [x;e;z] ∧ ¬(∃I∈tri_sides a b c. x∈I ∧ z∈I)"
```

4.2 Theorem 15 (i)

The first part of theorem 15 proves the existence of an intersection point assuming a configuration as shown in figure 4.4. This arrangement is important, since it appears in the proofs of both part (ii) and part (iii) as sub-configuration. Schutz states it as

> **Theorem 15 (i)** (Third Collinearity Theorem)
> *If abc and dbf are two kinematic triangles such that* $[afb]$ *and* $[bcd]$*, then* (ac) *meets* (df) *in an event e.* (Schutz [9, p. 31])

Its statement and proof does not make use of any new notions introduced in chapter 4, such as internal points or the boundary. Following Schutz's prose (sections A.2 and A.3 of the appendix), this theorem was successfully mechanized with only minor alterations. Its mechanization is therefore only outlined briefly here. Schutz begins by proving a lemma which captures a special case of part (i) as shown in figure 4.5, stated as follows

> **Lemma 1** (of Theorem 15)
> *For any events* a,b,c,d,f *such that there are paths* ab,ac,ad,bc,df *and* $[bcd], a \notin bc, [afb]$*, there is an event e such that* $[aec]$ *and* $[def]$.
>
> (Schutz [9, p. 31])

This was mechanized as:

```
lemma collinearity3_1_lemma:
  assumes
    pthsE: "[a;b]" "[a;c]" "[a;d]" "[b;c‖bc]" "[d;f]" and
    ord: "[b;c;d]" "[a;f;b]" and dist: "a ∉ bc"
  shows "∃e. [a;e;c] ∧ [d;e;f]"
```

Its proof was straightforward, with the single exception that Schutz at one point makes the unstated assumption (obvious from the figures) that an event obtained using Theorem 7 is the same as one that was already introduced. Since this issue occurred multiple times, the following auxiliary lemma was introduced which shows that an event obtained via the second collinearity theorem is the same as another event which satisfies either of several possible assumptions (compare figure 4.1 illustrating Theorem 7):

```
lemma collinearity2_events_equal:
assumes
  th7: "[a;f';b]" and "[d;e;f']" and
  path_ab: "ab∈𝒫" and "a∈ab" and "b∈ab" and "x∈ab"
  path_de: "de∈𝒫" and "d∈de" and "e∈de" and "x∈de" and
  paths_neq: "a∉de ∨ b∉de ∨ c∉ab ∨ d∉ab ∨ ab≠de"
shows "x = x'"
```

All of the possibilities in `paths_neq` effectively show the two paths `ab` and `de` are distinct. Using this auxiliary lemma, the tool *sledgehammer* was able to find a proof of equality in each case this problem arose. Using the above lemma, Schutz then goes on to proof part (i) of Theorem 15, which was mechanized as

```
theorem (*15*) collinearity3_1:
  assumes "△ a b c" and "△ d b f" and "[a;f;b]" and "[b;c;d]"
  shows "∃e. e ∈ segment a c ∧ e ∈ segment d f"
```

The mechanization was done following Schutz's proof with only minor alterations.

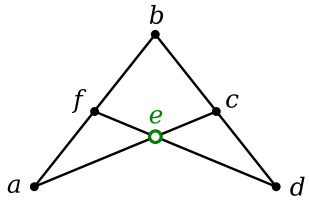

Figure 4.4: Theorem 15 (i).

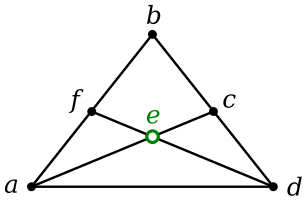

Figure 4.5: Lemma 1.

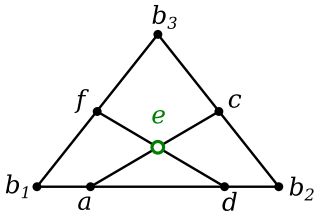

Figure 4.6: Lemma 2.

4.3 Theorem 15 (ii)

The second part of Theorem 15 was conceptually significantly harder to mechanize, both since it involves new notions whose definitions had to be chosen carefully, and

because the final bit of Schutz's proof required considerable unpacking. Schutz states the theorem as follows:

> **Theorem 15 (ii)** (Third Collinearity Theorem)
> *If two paths cross each other over a kinematic triangle, then they meet at an internal event.* (Schutz [9, p. 31])

Internal events were defined in section 4.1.3 and the definition for *cross over a kinematic triangle* is discussed in detail below. In order to prove part (ii), Schutz introduces a Lemma 2, which is visualized in figure 4.6, stated as follows:

> **Lemma 2** (of Theorem 15)
> *Let $b_1b_2b_3$ be a kinematic triangle and let ac, df be paths such that $[b_1fb_2]$, $[\![b_2adb_3]$, $[b_1cb_3]$. Then ac meets df in an event e such that $[aec]$ and $[def]$.* (Schutz [9, p. 31])

The remainder of Schutz's prose (section A.4 of the appendix) consists of a detailed proof of this lemma, which was mechanized in a straightforward way using the techniques developed for Lemma 1 in section 4.2 (see section B.2.4 of the appendix).

Schutz then claims Theorem 15 is "complete because all possible configurations occur as special cases of part (i) and Lemma 2". While this was eventually found to be true, it involved the conceptual difficulty of mechanizing the notion of *crossing over a triangle* and deducing the possible configurations such that part (i) and Lemma 2 can be applied. Schutz does not give a detailed explanation of this, and appears to rely on the pictorial representations and on the geometric intuitions of the reader instead.

In the following two sections, the mechanization of *crossing over a triangle* is developed and the involved design decisions discussed. Schutz introduces it as follows:

> A path which meets the boundary at exactly two distinct events separates the remaining subset of boundary events into two components: if a second path meets each of these components at exactly one event, we say that the two paths *cross each other over the kinematic triangle*.
> (Schutz [9, p. 31])

A visual representation of this is given in figure 4.8. The possible configurations (up to permutations) allowed by the above definition were derived by hand and are shown in figure 4.7. In section 4.3.1, we first mechanize the simpler notion of a single path crossing a triangle. Section 4.3.2 gives a summary of the different approaches to the final mechanized definition that were explored. After this, in section 4.3.3, the proof of Theorem 15 (ii) is concluded.

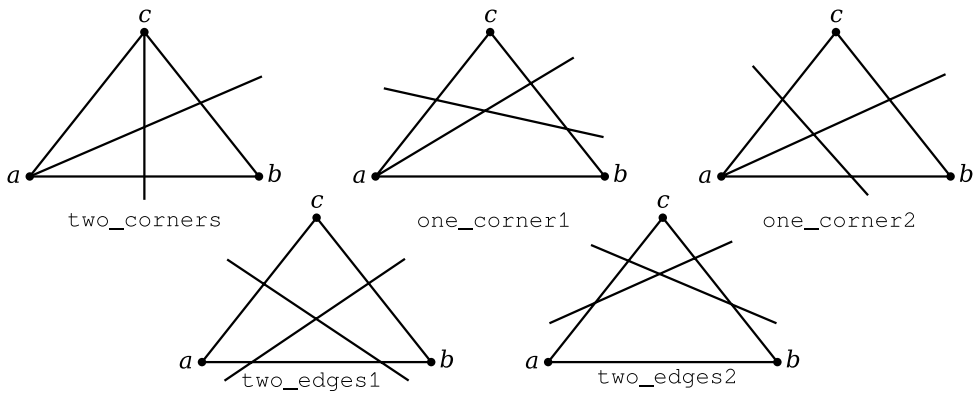

Figure 4.7: Possible configurations of two paths crossing over a triangle allowed by Schutz's definition and as derived on paper using pictorial representations.

4.3.1 Cross Kinematic Triangle

It will prove useful to define a predicate that states a path meets the boundary of a triangle at exactly two distinct events, as in Schutz's definition of *crossing over a triangle*, since this is what separates the boundary into two parts which the second path must meet. More importantly, it is used in the statement of Theorem 15 (iii) discussed in section 4.4. Following Schutz's, we must assert `card (P∩ℬ{a b c}) = 2`. We replace this with an equivalent expression that is more helpful for the proof in section 4.4.3, and define:

```
definition crosses_triangle :: ...
  where crosses_triangle a b c P ≡
    △ a b c ∧ P ∈ 𝒫 ∧ (∃x y. x ≠ y ∧ P∩ℬ{a b c} = {x, y})
```

The definition is automatically symmetric under permutations of the triangle corners, since both the boundary and the triangle predicate are.

This is a good opportunity to discuss a design decision which was made for this definition as well as the definitions of internal events and the boundary. While the predicate `internal_event a b c e` does not include the statement that `△ a b c`, the predicate above does and it also states `P∈𝒫`. One may argue that neither of these facts are strictly necessary, or that they should either consistently appear or be absent in both `internal_event` and `crosses_triangle`. The present difference between the two predicates is intentional and justified as follows.

The predicate `crosses_triangle` is meant to capture the complete geometric configuration of a path crossing a triangle and will never appear in a context where either `△ a b c` or `P∈𝒫` is false. While the latter may be said about `internal_event` as well, it becomes clear that it should not capture the complete geometric configuration

when written in the style `e ∈ ℐ{a b c}`. It seems unjustified for this to imply `△ a b c`, and it would be inconsistent with other similar notions (such as the boundary, or an interval in relation to a path).

It is also worth noting that the predicate `crosses_triangle` without the `△ a b c` and `P∈𝒫` reduced to `card (P∩ℬ{a b c}) = 2`. This would hardly justify introducing a dedicated definition.

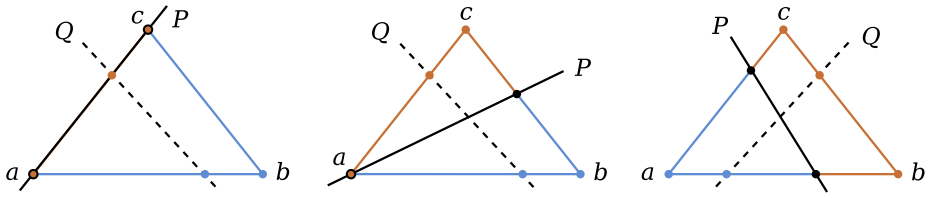

Figure 4.8: A path P crossing a triangle, splitting the boundary into two components, and a path Q touching each such that P and Q cross over the triangle. The left case can be excluded once Q is present or assuming denseness, since P would contain more than two boundary events.

4.3.2 Cross Over Triangle

The notion of two paths crossing over a triangle initially proved difficult to mechanize. Schutz's description suggests the mechanization should state that the boundary is split by a first path into two components, as illustrated in figure 4.8. Given this, it would be straightforward to introduce a second path in a way that captures the notion of crossing over the triangle. The goal is then to show that the definition implies a list of possible geometric arrangements as worked out on paper and shown in figure 4.7. This is necessary, since it amounts to the case distinction Schutz implicitly makes in his final step in proving Theorem 15 (ii).

An attempt to implement Schutz's definition literally is described in section 4.3.2.1. As this proved difficult to connect to more primitive notions of orderings, a unified definition which incorporates some pictorial intuition was developed, as summarized in section 4.3.2.2. Finally, a definition which incorporates the full pictorial intuition is introduced in section 4.3.2.3. With this, the final proof succeeded without further difficulties.

4.3.2.1 Splitting the Boundary

Initially, the goal was to define a predicate `boundary_split a b c P A B`, with the property that `A, B, P∩ℬ{a b c}` are a disjoint partition of `ℬ{a b c}` . Here, P is assumed to be crossing the triangle, i.e. `P∩ℬ{a b c} = {x,y}` for distinct `x`, `y`. The problem is immediately that the partition is unconstrained, which means `A` and `B` could have an arbitrary assignment of boundary events.

We clearly need to work harder to make `A` and `B` *components* in the sense Schutz means it (see figure 4.8). This leads to the notion of connectedness, as Schutz introduces it in Axiom I6 (connectedness of the unreachable set):

$$\texttt{∀r∈A. ∀t∈A. ∀s. [r;s;t]} \longrightarrow \texttt{s ∈ A.}$$

This constrains the possible partitions considerably. Neither set can contain events from both sides of a triangle edge which is split by the path. On paper, this seems to lead to the same configurations as shown in figure 4.8. It is not immediately clear, however, whether the constraint is sufficient to prove this rigorously in the axiomatic system (instead of pictorial reasoning). Initial efforts to prove this failed, partially because the techniques to handle without loss of generality assumptions (discussed in section 4.4.2) had not yet been developed and partially because the original definition was slightly faulty. In order not to waste time, we moved on to a second definition as outlined below and eventually found the inductive definition in section 4.3.2.3. Since the latter worked well for the proof of Theorem 15, we have not yet returned to this definition or proven equivalence.

4.3.2.2 Attempts at Unified Definition

Once a definition in terms of two sets representing the two components of the boundary was given up, the next attempt was to find a single definition for two paths crossing over a triangle directly. For this, we now consider two paths `P` and `Q` which each has two intersections with the boundary, `P∩ℬ{a b c}={p,q}` and `Q∩ℬ{a b c}={r,s}`.

If `P` splits the boundary into two components, we make the following observation. For all possible crossings over a triangle as shown in section 4.7, there is a corner v' of the triangle such that both r and s are *path connected within the boundary* to that corner. What this means pictorially is that one can draw an angle from r to the corner to s without leaving the boundary. This angle is the set $A = (rv') \cup \{v'\} \cup (v's)$. The property that the path `Q`, i.e. the set $\{p,q\}$ meets this set A exactly once is true if and only if the two paths cross over the triangle. This motivates the following definition:

```
abbreviation cross_over_tri_points :: ...
where "cross_over_tri_points a b c p q r s ≡ ∃v'∈{a,b,c}.
    interval r v' ⊆ ℬ{a b c} ∧
    interval v' s ⊆ ℬ{a b c} ∧
    card ({p, q} ∩ (interval r v' ∪ interval v' s)) = 1"
```

Crossing over a triangle would then involve the statement

$$\exists q{\neq}p.\ \exists r.\ \exists s{\neq}r.\ \mathcal{B}\{a\ b\ c\}\cap Q=\{p,q\}\ \wedge\ \mathcal{B}\{a\ b\ c\}\cap R=\{r,s\}\ \wedge$$
$$\texttt{cross\_over\_tri\_points a b c p q r s}$$

In the effort to prove that this implies combinations of figure 4.7, it emerged to be difficult (and potentially impossible at this point) to prove that `interval r v' ⊆ ℬ {a b c}` implies that r and v' are even on the same path. This means that the strong intuition of the points being path connected *within the boundary* was not obviously captured fully. It may be possible to prove this using the assumption of denseness, stating that $a,c \in \mathcal{P}$ implies there is an event b such that $[abc]$, which Schutz proves later in Theorem 17. However, even with this there was still much work needed to imply the derived possible configurations. For this reason the attempt at a unified definition was abandoned in favour of the inductive version introduced next.

4.3.2.3 Inductive Definition

In hindsight, it is quite likely that Schutz intended his introduction of two paths crossing over a triangle as a *description* rather than a *definition*. In this case, a formalization of his prose is misguided, as he expects the reader to think pictorially about all possibilities that conform to his description. This is possible since the components a crossing path splits the boundary into are intuitively clear. If we thus allow ourselves to unpack Schutz's description into the possible configurations using unproven intuition about the pictorial geometry, we can take these configurations as the definition.

In other words, instead of seeking some definitions close to Schutz's words, we define the proposition in terms of the possible configurations shown in figure 4.7 we are sure Schutz had in mind. This is done in Isabelle using an inductive definition, which lists all possible implications by which the proposition can be made true (see section B.2.1):

```
inductive cross_over_tri :: ... where
two_corners:
  "⟦ℬ{a b c}∩Q={a,x}; ℬ{a b c}∩R={c,y}; [b;x;c]; [a;y;b]...⟧
  ⟹ cross_over_tri a b c Q R" |
| one_corner1:
```

```
  "⟦ℬ{a b c}∩Q={a,x}; ℬ{a b c}∩R={r,s}; [a;s;c]; [b;r;x;c]...⟧
  ⟹ cross_over_tri a b c Q R" |
...
| cross_over_tri_sym:
  "⟦cross_over_tri a b c Q R; permut3 a b c a' b' c'⟧
  ⟹ cross_over_tri a' b' c' R Q"
```

In each case `...` indicates the repeated △ `a b c; Q∈𝒫; R∈𝒫`. With the last inductive case, we implicitly introduce all possible permutations of corners and paths, and also make the definition symmetric. The facts that `Q≠P` and each path meets the boundary twice are obvious from figure 4.7 and were proven for each case. These facts could have been included on the left hand sides of the inductive definitions, but minimal assumptions simplify proofs where it must be shown the predicate is true.

4.3.3 Proof of Theorem 15 (ii)

Using the inductive definition above, the second part indeed becomes as straightforward to prove as Schutz claims. The inductive definition can be used in the mechanized proof to show the final goal by proving it for each possible configuration of crossing over a triangle. It is easy to recognize the sub-configurations of Lemma 2 (figure 4.6) and part (i) (figure 4.4) in each of the inductive cases listed in figure 4.7. The mechanized proof is structured as follows:

```
theorem (*15*) collinearity3_2:
  assumes "cross_over_tri a b c Q R"
  (* Include additional result that will be useful later*)
  shows "∃x∈ℐ{a b c}. x∈Q ∧ x∈R
           ∧ path_spans_inner a b c Q x
           ∧ path_spans_inner a b c R x"
        (is "?IntE a b c Q R")
proof -
  have "cross_over_tri a b c Q R ⟹ ?IntE a b c Q R"
  proof (induction rule: cross_over_tri.induct)
    case (two_corners a b c Q R x y) ...
  next ... qed
  thus ?thesis using assms by auto
qed
```

Note that the result of the lemma was extended compared to Schutz's statement to include `path_spans_inner a b c Q x` and `path_spans_inner a b c R x`. These are results which will become essential in the proof of the second part of Theorem 15 discussed below. The predicate `path_spans_inner` captures that `x` is between the two events where the path crosses the boundary:

```
abbreviation path_spans_inner :: ...
  where "path_spans_inner a b c P e ≡
    ∃x z. 𝓑{a b c}∩P = {x, z} ∧ [x;e;z]"
```

This result was a direct implication of facts that were already available in each case.

4.4 Theorem 15 (iii)

The third part of Theorem 15 is in some sense the converse of the second part. Schutz states it as:

> **Theorem 15 (iii)** (Third Collinearity Theorem)
> *Two paths which meet at an internal event (and which meet the boundary at four distinct events), cross each other over the kinematic triangle.*
>
> (Schutz [9, p. 31])

Schutz simply claims that this is a direct implication of previous results:

> Part (iii) is an immediate consequence of part (ii), the Second Collinearity Theorem (Th.7) and Theorem 8. (Schutz [9, p. 35])

This is the full extent of the proof which he gives. Reviewing Theorems 7 (figure 4.1) and 8 (figure 4.2), it appears that we must rigorously derive all possible arrangements in which two paths may cross a triangle and lead all cases where they do not cross over the triangle (as defined above), i.e. they pass each other, to a contradiction using the assumptions. In contrast to part (ii) above, we are not at liberty to introduce an inductive definition which immediately gives us all possible configurations (which we could derive on paper). We must instead start from Schutz's assumptions above and deduce the possible configurations. Doing this pictorially and on paper is not very difficult. The paths either cross, which are the configurations shown in figure 4.7, or they pass each other in one of the arrangements shown in figure 4.9.

In summary, there are two parts to the proof Theorem 15 (iii) which Schutz seems to imply. First, we need to deduce the possible configurations allowed given the assumptions of the theorem. Then, we must lead all configurations in which the paths do not cross over the triangle to a contradiction. In section 4.4.1 below, a reformulation of the theorem (later proved equivalent) is discussed and the proof we surmise Schutz to have meant is outlined. We will see that the proof is relatively straightforward if we make an intuitive assumption about the internal event, namely that internal events are always *inner* events as described below. This leads to a lemma we need to proof, referred to in the following as `internal_always_inner_event`.

After the proof outline, the general method of WLOG (without loss of generality) lemmas, used to avoid combinatorial explosions when considering possible arrangements, is introduced in section 4.4.2, together with a proof that internal events are not on the boundary (this will be used in proving `internal_always_inner_event`). Before moving to two paths, as in the theorem, we first rigorously derive possible ways a single path may cross a triangle in section 4.4.3. The mechanized, rigorous deduction of possible arrangements for two paths is discussed in 4.4.4.

Once this construction is completed, we unpack `internal_always_inner_event` into several layers of results, each of which required a careful consideration of all possible configurations and permutations. While Schutz may have taken these to be obvious, following from pictorial representations, we annot use this argument and must provide rigorous proofs. Finally, the proof of part (iii) is outlined in section 4.4.6.

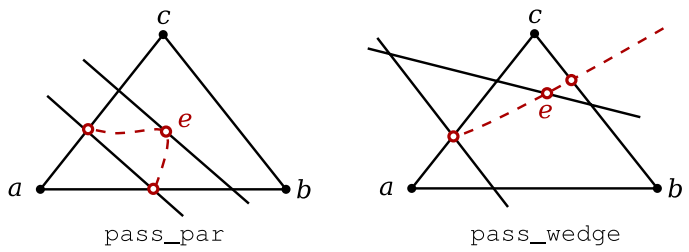

Figure 4.9: Two paths crossing a triangle and passing by each other. In red, the contradiction with Theorem 7 (on the right) and Theorem 8 (on the left) is illustrated, assuming the paths meet at the internal point e.

4.4.1 Outline of Proof

We start here from a slightly modified version of part (iii), which is shown to imply Schutz's literal version in section 4.4.6. The following is written in prose style with references to the corresponding mechanizations. We intend to show

Theorem 15 modified (iii)
Two distinct paths which both cross a kinematic triangle and meet each other at an internal point, cross each other over the kinematic triangle.

Proof. First, observe that the two paths cannot meet each other on the boundary. Assuming they do, note that they would meet twice, since the internal event is not on the boundary (this is presumably assumed by Schutz as obvious; proven rigorously

as `internal_event_not_boundary` in section 4.4.2). Then the two paths are equal, contradicting the assumption.

Next, we consider the possible arrangements in which the two paths may intersect the boundary (see section 4.4.4). Either the two paths cross over the triangle, in which case the proof is complete, or they miss each other.

Without loss of generality (this is derived using `two_crosses_cases` in section 4.4.4), there are two arrangements, shown in figure 4.9, in which the paths can pass each other. Either both pass through the same two sides of the triangle (`pass_par`) or only one side is intersected by both (`pass_wedge`). We can immediately lead each case to a contradiction using the fact that no path meets each side of a triangle and the second collinearity theorem, respectively, if we assume the following intuitive property.

Lemma `internal_always_inner_event` **(**Internal event always between crossings**)** *Consider a path P that crosses the triangle* $\triangle$ *a b c and meets the boundary at events x and z. For any internal event e which is also in P, we must have* $[xez]$*. With the predicate introduced earlier, we always have* `path_spans_inner a b c P e`*.*

This lemma follows directly from our intuitive understanding of "internal", since an ordering $[xze]$ would put the internal event outside of the triangle. Proving it rigorously in the axiomatic system was a considerable challenge and involved proving several intermediate results. This is discussed in section 4.4.5.2 The contradictions arising in each case using this lemma are discussed in more detail in section 4.4.5.1. □

4.4.2 Internal Events Are Not on Boundary

Schutz appears to make several geometrically intuitive assumptions. One such assumption which Schutz does not prove is that given a triangle $\triangle$ `a b c` an internal event `e`$\in\mathcal{I}$`{a b c}` is not also on the boundary, `e`$\notin\mathcal{B}$`{a b c}`. A mechanized proof of this statement was constructed and is briefly introduced here.

Given the definition of `internal_event` in section 4.1.3, we know that we must consider the two possible arrangements shown in figure 4.3. Unfortunately, these arrangements may appear in any possible permutation of corners. We thus want to tell Isabelle that, without loss of generality, we consider the permutation in which one of the two arrangements is true. This is possible because both assumption $\triangle$ `a b c` and proposition `e`$\notin\mathcal{B}$`{a b c}` are symmetric with respect to *a*, *b*, *c*. Isabelle can be told of the WLOG proof method rule using the following lemma:

```
lemma internal_event_wlog:
  assumes asm_sym: "⋀ a b c a' b' c'.
    ⟦A a b c; permut3 a b c a' b' c'⟧ ⟹ A a' b' c'"
  and pro_sym: "⋀ a b c a' b' c'.
    ⟦P a b c; permut3 a b c a' b' c'⟧ ⟹ P a' b' c'"
  and wlog_case: "⋀ a b c x z.
    ⟦A a b c e; [a;x;b]; [x;e;z]; [b;z;c]∨z=c⟧ ⟹ P a b c e"
  shows "⋀ a b c.
    ⟦A a b c; internal_event a b c e⟧ ⟹ P a b c"
```

The first assumption introduces some unspecified predicate `A` and states that this must be symmetric under permutations. The second assumption introduces another unspecified predicate `P` which is also symmetric. The third assumption states that `A` together with x, z and e as in the two possible arrangements in the definition of the internal event implises `P`. This is claimed to show that `A a b c` together with `internal_event a b c e` implies `P a b c`. In other words, assuming the general predicates `A` and `P` are symmetric, the goal of showing `A` implies `P` given an internal event `e` can be replaced with proving `P` for one permutation, and all other permutation follow automatically.

This rule can now be used to prove the original statement:

```
lemma internal_event_not_boundary:
  assumes "△ a b c" "internal_event a b c e"
  shows "e ∉ ℬ{a b c}"
  proof(rule internal_event_wlog[where
          A="λ a b c. △ a b c" and
          P="λ a b c. e ∉ ℬ{a b c}"], rule_tac[4-5] assms)
    show ... (* Assumption and Proposition symmetric *)
  next
    fix a b c x z
    assume "△ a b c" "[a;x;b]" "[x;e;z]" "[b;z;c] ∨ z = c"
    show "e ∉ ℬ{a b c}" ...
```

For Isabelle to understand how to apply `internal_event_wlog` as a rule, we specify what the predicates `A` and `P` are using the **`where`** keyword.

The crucial step in the above proof, which would appear after line 13, is to consider each position on the boundary `e` could be located at, and leading each case to a contradiction. This is done using the fact that two distinct events specify a path uniquely, and each possible position of `e` implies two sides of the triangle are equal, which is by definition false.

The result above can be extended to the paths spanning a side of the triangle:

```
lemma internal_not_on_tri_path:
  assumes "△ a b c" "internal_event a b c e" "[a;b‖ab]"
  shows "e ∉ ab"
```

4.4.3 Cases of One Path Crossing a Triangle

We eventually aim to show possible configurations of two paths crossing a triangle, as required for the case distinction in the proof of part (iii). We first show what configurations are possible for a single path crossing a triangle. For a single path crossing a triangle, we know that it touches the boundary exactly twice. We consider three cases: Two of the intersections may be corners, one of them may be a corner or neither of them may be a corner. In Isabelle, the lemma looks as follows:

```
lemma crosses_tri_cases:
  assumes "crosses_triangle a b c P"
  shows "∃a' b' c'. permut3 a b c a' b' c' ∧ (
         ℬ{a' b' c'} ∩ P = {a', b'} ∨
         (∃x. [b';x;c'] ∧ ℬ{a' b' c'}∩P={a',x}) ∨
         (∃x y. [a';x;b'] ∧ [c';y;a'] ∧ ℬ{a' b' c'}∩P={x,y}))"
proof -
  ...
  obtain x y where "x ≠ y" and "ℬ{a b c}∩P = {x, y}" <proof>
  then consider
      (card2) "card ({x,y}∩{a,b,c}) = 2" |
      (card1) "card ({x,y}∩{a,b,c}) = 1" |
      (card0) "{x,y}∩{a,b,c} = {}"
    <proof>
  thus ?thesis
  proof (cases) ... qed
qed
```

The possible configurations claimed to be implied by `crosses_triangle` in this lemma are visualized in figure 4.8. The first case amounts to the path P being aligned with one of the edges and is spurious. It is only included because at this point we cannot show that the interval $|ab|$ contains more than two events. If it does, the third event would contradict the assumption that the path touches the boundary exactly once. Luckily, all of these spurious configurations eventually lead to contradictions once we have more than one path crossing the triangle, which allows Theorem 15 to be proved without assuming denseness.

Two kinds of WLOG lemmas are introduced (section B.2.2 in the appendix). One in which the corners of the triangle can be permuted without loss of generality, and another where this is not assumed to be possible.

4.4.4 Cases of Two Paths Crossing a Triangle

Given the two WLOG lemmas single paths, it is now relatively straightforward to show what configurations are possible for two paths. While for the first path we can

permute the corners of the triangle freely, once its intersection with the triangle is fixed, we cannot repeat the same trick (the assumptions are no longer symmetric). Instead, we use the second WLOG lemma introduced above, which does not permute the corners. In addition to crossing the triangle, we also assume the paths do not meet on the boundary. We show the following lemma, which claims that either the two paths cross over the triangle, or there is a permutation of corners such that one of the configurations shown in figure 4.9 holds:

```
lemma two_crosses_cases:
  assumes "crosses_triangle a b c Q"
      and "crosses_triangle a b c R" and "Q∩R∩ℬ{a b c}={}"
  shows "cross_over_tri a b c Q R ∨
     (∃a' b' c'. permut3 a b c a' b' c' ∧ (
       (∃p q r s. ℬ{a' b' c'}∩Q={p,q} ∧ ℬ{a' b' c'}∩R={r,s}
                ∧ [a';p;r;b']] ∧ [a';q;s;c']])
     ∨ (∃p q r s. ℬ{a' b' c'}∩Q={p,q} ∧ ℬ{a' b' c'}∩R={r,s}
                ∧ [a';r;p;b']] ∧ [a';s;q;c']])
     ∨ (∃p q r s. ℬ{a' b' c'}∩Q={p,q} ∧ ℬ{a' b' c'}∩R={r,s}
                ∧ [a';p;b'] ∧ [b';r;c'] ∧ [a';q;s;c'])))"
```

The proof for each case is relatively straight-forward. Most of the spurious alignments with sides can be shown to lead to contradictions, unless the two paths pass each other. The other configurations are either shown to correspond to one of the defining configurations of `cross_over_tri` (or one of their permutations), or to one of the cases where the paths do not meet over the triangle. Given this lemma, it is straightforward to prove corresponding WLOG lemmas (see section B.2.3 of the appendix).

4.4.5 Internal Event Ordering

The goal of this section is to prove the lemma `internal_always_inner_event` in order to complete the proof of Theorem 15 (iii). It states that any path P which crosses the boundary of a triangle twice at points x and z, and contains an internal point e, satisfies $[x;e;z]$. In order to show this, we need to use the definition of internal points. First, in section 4.4.5.1, lead all configurations to contradictions where two paths cross a triangle and pass each other, but one of them has an internal event ordered as above. We use this in section 4.4.5.2 To lead up to the result of the lemma.

4.4.5.1 Passing Paths Share No Internal Point

As an important ingredient in showing both lemma `internal_always_inner_event`, and part (iii) of Theorem 15 itself, we show that two paths that do not cross over a

triangle (i.e. pass each other) cannot share an internal event if one of paths satisfies `path_spans_inner a b c Q x`. This last assumption is critical as it captures the definition of an internal event. The motivation for formulating this lemma exactly like this will become clearer in the next section. The first lemma regards the configuration `pass_wedge` of figure 4.9 where the outer of the two paths satisfies `path_spans_inner`:

```
lemma pass_par_no_internal_outer:
  assumes "△ a b c" "Q ∈ 𝒫" "R ∈ 𝒫"
          "ℬ{a b c} ∩ Q = {p, q}" "ℬ{a b c} ∩ R = {r, s}" and
    meet: "[r;e;s]" "e∈R" "e∈Q" and
    order: "[a;p;r;b]]" "[a;q;s;c]]"
  shows False
```

The proof of this is straightforward using Theorem 8 and consists of establishing the facts required for the application of this theorem.

The second lemma `pass_par_no_internal_inner` is almost the same as above, the ordering is known for the inner path Q instead of R. If the same ordering applies to the outer path, we can get a contradiction via the previous lemma. The other two possible orderings $[p\,q\,e]$ and $[e\,p\,q]$ are symmetric under exchange of p and q, and can both be lead to a contradiction using Theorem 7. For the final lemma, we consider the configuration `pass_wedge` from figure 4.9.

```
lemma pass_wedge_no_internal:
  assumes "△ a b c" "Q ∈ 𝒫" "R ∈ 𝒫"
          "ℬ{a b c} ∩ Q = {p, q}" "ℬ{a b c} ∩ R = {r, s}" and
    meet: "[r;e;s]" "e∈R" "e∈Q" and
    order: "[a;p;b]" "[b;r;c]" "[a;q;s;c]"
  shows False
```

The proof again makes use of Theorem 7, by which the path Q can be shown to touch every segment of the triangle boundary, which contradicts Theorem 8.

4.4.5.2 Internal Event is Always Inner Event

We now want to show the lemma `internal_always_inner_event` which was introduced as the intuitive fact used to prove part (iii) in section 4.4.1. Given a path P which crosses $\triangle\, a\, b\, c$ and meets the boundary at x and z, we want to show that $[x\,e\,z]$ for any internal event e. It is inevitable that we must make use of the definition of internal events. As a first step, therefore, the following lemma is proved.

```
lemma internal_event_get_crosses_tri:
  assumes "△ a b c" "internal_event a b c e"
  shows "∃P. P∈𝒫 ∧ crosses_triangle a b c P
              ∧ path_spans_inner a b c P e"
```

We can obtain the path P directly using the definition of internal events and Axiom O1. The only thing left to show is that there is no third event which is both on P and on the boundary of the triangle. This can be proved easily, as the assumption of such a path either immediately implies e is on the boundary, or is the case considered in `pass_par_no_internal_outer`. The next step is showing the following lemma:

```
lemma internal_both_inner_event:
  assumes cross: "crosses_triangle a b c Q"
                 "crosses_triangle a b c R"
                 "Q∩R∩ℬ{a b c}={}"
      and meet: "ℬ{a b c}∩Q={p,q}" "ℬ{a b c}∩R={x,z}"
                "[x;e;z]" "e∈R" "e∈Q"
    shows "[p;e;q]"
```

The statement here is that `path_spans_inner` can be transferred from one path to the other if the paths do not meet on the boundary but meet in the internal point for which one ordering is known. To prove this, we use the WLOG lemma for two paths crossing a triangle. In the case of the path crossing over the triangle, we are done using the additional result that was included in part (ii), which states exactly the desired property. The cases of the paths passing each other led to contradictions using the results of the previous section. Finally, using the above results, the final lemma follows:

```
lemma internal_always_inner_event:
  assumes "crosses_triangle a b c Q"
      and "ℬ{a b c}∩Q={p,q}" and "e ∈ Q∩ℐ{a b c}"
  shows "[p;e;q]"
```

4.4.6 Proof of Part (iii)

Since the lemma `internal_always_inner_event` was proved above, we can now mechanize a proof for the modified version of part (iii) following the proof-sketch of section 4.4.1. The mechanized version of the modified part (iii) is

```
theorem (*15*) collinearity3_3:
assumes "crosses_triangle a b c Q" "crosses_triangle a b c R"
    and "Q ≠ R" "e ∈ Q ∩ ℐ{a b c}"
shows "cross_over_tri a b c Q R"
```

Given this, it is straightforward to prove a mechanized version of Schutz's formulation also (see section B.2.5 in the appendix):

```
theorem (*15*) collinearity3_3':
  assumes "△ a b c"
      and "Q∈𝒫" "R∈𝒫" "e ∈ Q ∩ R ∩ ℐ{a b c}"
      and disj: "card ((Q ∪ R) ∩ ℬ{a b c}) = 4"
    shows "cross_over_tri a b c Q R"
```

Chapter 5

Conclusions

The mechanization of Schutz's axiomatic system of Minkowski space was successfully advanced. An initial critical reading of the existing mechanizations led to improvements in notation and significant reduction of complexity in several parts, most notably in the definition of chains. Theorem 15 was mechanized, which involved formalizing Schutz's definitions and filling in large parts of pictorial or intuitive reasoning.

Although not fully unexpected, it was surprising to find that The last two sentences of Schutz proof to Theorem 15 proved the most difficult and most time-consuming to mechanize. As mentioned, the reasons for this were that some of Schutz's notions could not be mechanized easily by following his prose literally, and that he seems to have assumed geometrically obvious steps that turned out difficult to prove rigorously. This also involved some advanced efforts in using WLOG lemmas to simplify proofs, since a naive case distinction would have led to a combinatorial explosion that would have been tedious and time consuming to manage.

It is worth noting that Schutz claims part (iii) is the immediate consequence, among other facts, of part (ii). We have used part (ii) in our proof only for the fact that an internal event e is ordered like $[xez]$ for all events x and z that share a path with e, which we suspect Schutz may have taken for granted. Notably, this was only possible after we extended part (ii) by the result that the obtained internal event has such an ordering, which Schutz did not specify even though it is an immediate result of his proof. This seems to suggest either that Schutz did take the ordering of internal events for granted, in which case part (ii) would not have been used at all, or he may have had another or slightly different proof in mind. It is possible that this alternative proof may have been slightly easier to formalize, although it would likely have involved a similar kind of reasoning. The final theorem implies the intuitive facts we used in our

proof , so it is likely that we have not shown more results than required, and whatever alternative proof may exist would likely not be significantly simpler.

Although no fixed goal was set for the effort of continuing the formalization into chapter 4, the general expectation was to mechanize more than one theorem. In this regard one must acknowledge that Schutz's theorems become progressively more difficult, building on previous results, and involving more complex geometric arrangements. A mechanization is made more difficult especially by the fact that Schutz uses pictorial reasoning, which cannot be easily translted to Isabelle.

Finally, we remark that our inductive definition of two paths crossing over a triangle was nicely validated by the fact that two paths crossing a triangle were rigorously shown to either pass each other, or cross over the triangle by our definition.

5.1 Future work

There are several theorems of chapter 4 left to formalize. In Theorem 16, Schutz introduces an order topology on paths. Here, it may be worth investigating whether existing theories about order topologies in Isabelle could be reused or whether new definitions are needed. An investigation of this could unfortunately not be concluded in this project. As mentioned in section 3.3, it is likely possible to reorganize the mechanization more carefully into locales. Some work in this direction has been done in this project, but needed to be postponed in order to finish the proof of Theorem 15.

It was found that often, many needed facts can be just read off from a pictorial representation while requiring tedious derivation in Isabelle. A tool to generate facts from existing ones using a specified set of rules would be convenient, including e.g. permutations of facts or the existence of paths and kinematic triangles. Some work in this direction appears to have been done [36], but was not investigated for this project.

It would, furthermore, be very useful if one could generate pictorial representations of a given set of properties (triangles, intersecting paths, etc.) from a given set of statements or for a lemma. This would help both in the development of proofs, as one can check the meaning of variables more quickly and see what facts are available, as in the reading of the results themselves.

Lastly, we note that while WLOG lemmas were indispensable for the proof of Theorem 15 (iii), they were rather tedious to prove even though their proves mainly exploit symmetry properties. This suggests it may be possible to design general algorithms which can derive WLOG results or apply them automatically as a proof method.

Bibliography

[1] A. Einstein. “Zur Elektrodynamik bewegter Körper”. In: *Annalen der Physik* 322.10 (1905), pp. 891–921. ISSN: 0003-3804. DOI: 10.1002/andp.19053221004.

[2] Steven Weinberg. *Gravitation and cosmology: principles and applications of the general theory of relativity*. New York: Wiley, 1972. 657 pp. ISBN: 978-0-471-92567-5.

[3] Charles W. Misner. *Gravitation*. In collab. with Kip S. Thorne and John Archibald Wheeler. San Francisco, Calif.: WHFreeman, 1973. ISBN: 978-0-7167-0334-1.

[4] Hermann Minkowski. “Die Grundgleichungen für die elektromagnetischen Vorgänge in bewegten Körpern”. In: *Mathematische annalen* 68.4 (1910), pp. 472–525. ISSN: 0025-5831. DOI: 10.1007/BF01455871.

[5] A. Einstein and J. Laub. “Über die elektromagnetischen Grundgleichungen für bewegte Körper”. In: *Annalen der Physik* 331.8 (1908), pp. 532–540. ISSN: 0003-3804. DOI: 10.1002/andp.19083310806.

[6] Robin Hartshorne. *Geometry: Euclid and beyond*. Corr. second printing. Undergraduate texts in mathematics. New York, London: Springer, 2002. ISBN: 978-0-387-98650-0.

[7] John W. Schutz. “An axiomatic system for Minkowski space–time”. In: *Journal of mathematical physics* 22.2 (1981), pp. 293–302. ISSN: 0022-2488. DOI: 10.1063/1.524877.

[8] John W. Schutz. *Foundations of Special Relativity: Kinematic Axioms for Minkowski Space-Time*. Vol. 361. Lecture Notes in Mathematics. Berlin, Heidelberg: Springer, 1973. ISBN: 978-3-540-06591-3.

[9] John W. Schutz. *Independent axioms for Minkowski space-time*. Pitman research notes in mathematics series 373. Harlow: Longman, 1997. ISBN: 978-0-582-31760-4.

[10] Jake Evan Palmer. "Formal axiomatisation of Minkowski spacetime". MSc Dissertation. Edinburgh, UK: University of Edinburgh, 2017. URL: https://project-archive.inf.ed.ac.uk/all/msc/20172520/msc_proj.pdf.

[11] Jake Palmer and Jacques Fleuriot. "Mechanising an Independent Axiom System for Minkowski Space-time". In: 12th International Conference on Automated Deduction in Geometry. Nanning, China, Sept. 2018, pp. 64–79.

[12] Richard Schmoetten. "Axiomatic Minkowski Spacetime in Isabelle/HOL". MSc Dissertation. Edinburgh, UK: University of Edinburgh, 2020. URL: https://project-archive.inf.ed.ac.uk/msc/20204462/msc_proj.pdf.

[13] Thomas Little Heath. *The Thirteen Books of Euclid's Elements*. Courier Corporation, 1956.

[14] David Hilbert. *Grundlagen der Geometrie*. 7. umgearbeitet und vermehrte Auflage. Leipzig und Berlin: BGTeubner, 1930.

[15] A. N. Gorban. "Hilbert's sixth problem: the endless road to rigour". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376.2118 (Apr. 28, 2018), p. 20170238. DOI: 10.1098/rsta.2017.0238.

[16] Alfred Arthur Robb. *Geometry of Time and Space*. Cambridge University Press, 1936.

[17] Brent Mundy. "The Physical Content of Minkowski Geometry". In: *British Journal for the Philosophy of Science* 37.1 (1986), pp. 25–54. DOI: 10.1093/oxfordjournals.bjps/37.1.25.

[18] Oswald Veblen. "A System of Axioms for Geometry". In: *Transactions of the American Mathematical Society* 5.3 (1904), pp. 343–384. ISSN: 00029947.

[19] Robert L. Moore. "Sets of Metrical Hypotheses for Geometry". In: *Transactions of the American Mathematical Society* 9.4 (1908), pp. 487–512. ISSN: 00029947. URL: http://www.jstor.org/stable/1988666.

[20] A. G. Walker. "Axioms for cosmology". In: *Studies in Logic and the Foundations of Mathematics*. Vol. 27. Elsevier, 1959, pp. 308–321.

[21] George Szekeres. "Kinematic geometry; an axiomatic system for Minkowski space-time: ML Urquhart in memoriam". In: *Journal of the Australian Mathematical Society* 8.2 (1968), pp. 134–160.

[22] Robert Goldblatt. *Orthogonality and Spacetime Geometry*. Universitext. New York: Springer-Verlag, 1987. ISBN: 978-0-387-96519-2. DOI: 10.1007/978-1-4684-6345-3.

[23] Robert Goldblatt. "First-Order Spacetime Geometry". In: *Logic, Methodology and Philosophy of Science VIII*. Ed. by Jens Erik Fenstad, Ivan T. Frolov, and Risto Hilpinen. Vol. 126. Studies in Logic and the Foundations of Mathematics. Elsevier, 1989, pp. 303–316. DOI: 10.1016/S0049-237X(08)70051-X.

[24] Hajnal Andréka et al. "On Logical Analysis of Relativity Theories". In: (2011).

[25] Mike Stannett et al. "Using Isabelle/HOL to Verify First-Order Relativity Theory". In: *Journal of automated reasoning* 52.4 (2014), pp. 361–378. ISSN: 0168-7433. DOI: 10.1007/s10817-013-9292-7.

[26] Lawrence C. Paulson. *Isabelle - A Generic Theorem Prover (with a contribution by T. Nipkow)*. Vol. 828. Lecture Notes in Computer Science. Springer, 1994. ISBN: 3-540-58244-4. DOI: 10.1007/BFb0030541.

[27] Ernst Zermelo. "Über grenzzahlen und mengenbereiche: Neue untersuchungen über die grundlagen der mengenlehre". In: *Fundamenta mathematicae* 16 (1930).

[28] Lawrence C. Paulson and Krzysztof Grabczewski. "Mechanizing set theory". In: *Journal of Automated Reasoning* 17.3 (Dec. 1, 1996), pp. 291–323. ISSN: 1573-0670. DOI: 10.1007/BF00283132.

[29] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. 2nd ed. Applied Logic Series. Springer Netherlands, 2002. ISBN: 978-1-4020-0763-7. DOI: 10.1007/978-94-015-9934-4. (Visited on 08/12/2021).

[30] Michael Gordon, Robin Milner, and Christopher Wadsworth. "Edinburgh LCF: a mechanised logic of computation". In: *Lecture Notes in Computer Science* 78 (1979).

[31] Lawrence C. Paulson, Tobias Nipkow, and Makarius Wenzel. "From LCF to Isabelle/HOL". In: *Formal Aspects of Computing* 31.6 (Dec. 1, 2019), pp. 675–698. ISSN: 1433-299X. DOI: 10.1007/s00165-019-00492-1.

[32] Dana S Scott. "A type-theoretical alternative to ISWIM, CUCH, OWHY". In: *Theoretical Computer Science* 121.1-2 (1993), pp. 411–440.

[33] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*. Vol. 2283. Lecture Notes in Computer Science. Springer, 2002. ISBN: 3540433767. DOI: 10.1007/3-540-45949-9.

[34] Lawrence C. Paulson and Jasmin Christian Blanchette. "Three years of experience with Sledgehammer, a Practical Link Between Automatic and Interactive Theorem Provers". In: *EPiC Series in Computing*. IWIL 2010. The 8th International Workshop on the Implementation of Logics. Vol. 2. EasyChair, May 15, 2012, pp. 1–11. DOI: 10.29007/36dt.

[35] Markus Wenzel. "Isabelle/Isar — a versatile environment for human-readable formal proof documents". PhD thesis. Munich, Germany: Technische Universität München, 2002. URL: https://isabelle.in.tum.de/Isar/isar-thesis-Isabelle2002.pdf (visited on 04/16/2021).

[36] Phil Scott and Jacques Fleuriot. "An Investigation of Hilbert's Implicit Reasoning through Proof Discovery in Idle-Time". In: *Automated Deduction in Geometry: 8th International Workshop, ADG 2010, Munich, Germany, July 22-24, 2010, Revised Selected Papers* (2011), pp. 182–200. DOI: 10.1007/978-3-642-25070-5_11. (Visited on 08/13/2021).

Appendix A

Original Text (Chapter 4, Schutz 1997)

For reference, relevant parts of Schutz's text [9] are given below.

A.1 Third Collinearity Theorem

4.1 Third collinearity theorem

Given a kinematic triangle $\triangle a_1 a_2 a_3$, the three intervals $|a_1 a_2|$, $|a_2 a_3|$, $|a_3 a_1|$ are called *sides* and their union is called the *boundary* $\mathcal{B}(a_1, a_2, a_3)$. Any event between two events of distinct sides of the triangle is called an *internal event*.

A path which meets the boundary at exactly two distinct events separates the remaining subset of boundary events into two components: if a second path meets each of these components at exactly one event, we say that the two paths *cross each other over the kinematic triangle*.

Theorem 15 (Third Collinearity Theorem)

(i) *If abc and dbf are two kinematic triangles such that* $[afb]$ *and* $[bcd]$, *then* (ac) *meets* (df) *in an event e* (see Figure 9).

(ii) *If two paths cross each other over a kinematic triangle, then they meet at an internal event.*

(iii) *Two paths which meet at an internal event (and which meet the boundary at four distinct events), cross each other over the kinematic triangle.*

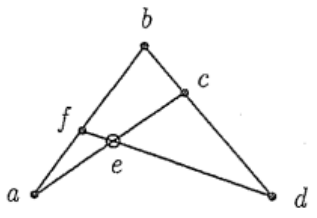

Figure 9

A.2 Lemma 1 - For Proof of Part (i)

Remarks Part (i) is a converse result to the Second Collinearity Theorem (Th.7). We will first prove Lemma 1 and then part (i). Next we will prove Lemma 2 followed by parts (ii) and (iii).

Lemma 1 *For any events* a, b, c, d, f *such that there are paths* ab, ac, ad, bc, df *and*

$$[bcd]\ , \quad a \notin bc\ , \quad [afb]$$

there is an event e *such that* $[aec]$ *and* $[def]$ (see Figure 10a).

Remarks This lemma is almost a converse of Theorem 7 except that here we postulate the existence of the path ad. Lemma 1 will be superseded by (i) (of this Theorem 15) which is converse to Theorem 7.

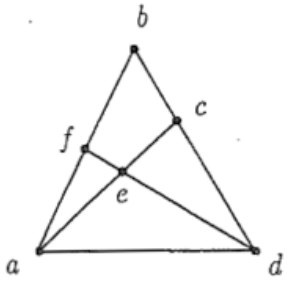

Figure 10a

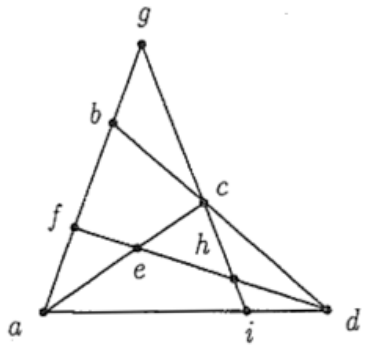

Figure 10b

Proof (of Lemma 1 which is based on Veblen (1904), Theorem 13, p358).
The preceding theorem (Th.14(ii)) implies the existence of an event g and a path cg such that $[afbg]$ (as in Figure 10b). Now the Second Collinearity Theorem (Th.7) applied to the kinematic triangle $\triangle dfb$ with $[fbg]$ and $[bcd]$ implies the existence of an event h such that $[dhf]$ and $[gch]$. Similarly for the kinematic triangle $\triangle daf$ with $[afg]$ and $[fhd]$ there is an event i such that $[dia]$ and $[ghi]$, and then Theorem 10 implies $[gchi]$. Another application of the Second Collinearity Theorem to the kinematic triangle $\triangle cai$ with $[aid]$ and $[ihc]$ implies the existence of an event e such that $[cea]$ and $[dhe]$. A final application of the same theorem to the kinematic triangle $\triangle abc$ with $[bcd]$ and $[cea]$ implies that ($[aec]$ and) $[def]$ which completes the proof of Lemma 1.

A.3 Proof of Part (i)

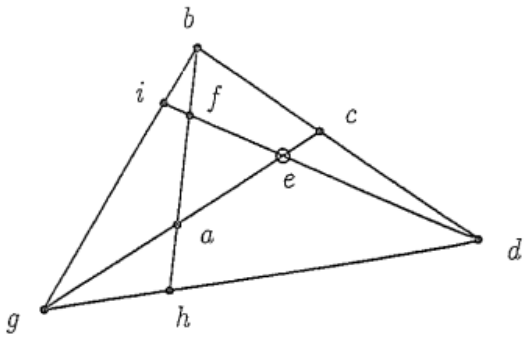

Figure 11

Proof (of (i)). The Second Existence Theorem (Th.14) implies the existence of an event g and paths gb, gd such that $[gac]$ (Figure 11). Then the Second Collinearity Theorem (Th.7) applied to the kinematic triangle $\triangle gdc$ with $[dcb]$ and $[cag]$ implies the existence of an event h such that $[ghd]$ and $[bah]$, whence Theorem 10 implies $[bfah]$. Again, the Second Collinearity Theorem applied to the triangle $\triangle bgh$ with $[ghd]$ and $[hfb]$ implies that df meets bg in an event i such that $[big]$. Finally Lemma 1 (above) (with g, b, c, d, i taking the place of a, b, c, d, f respectively) implies the existence of an event e such that $[gec]$ and $[dei]$.

Now Theorem 10 with $[gac]$ and $[gec]$ implies that either: (α) $a = e$, (β) $[geac]$ or (γ) $[gaec]$. Case (α) would imply that $a(= e) = f$ which would contradict $[bfah]$. Case (β) with the Second Collinearity Theorem applied to the kinematic triangle $\triangle gha$ with $[haf]$ and $[aeg]$ gives $[gdh]$ which contradicts the previously obtained $[ghd]$. The only remaining possibility is Case (γ) which implies $[aec]$. Since the statement of the theorem is symmetric with respect to interchange of the symbols a with d and f with c, the second order relation $[def]$ can be established in a similar manner: this completes the proof of (i).

A.4 Lemma 2 - For Proof of Part (ii)

Lemma 2 *Let $b_1b_2b_3$ be a kinematic triangle and let ac, df be paths such that $[b_1fb_2]$, $[b_2adb_3]$, $[b_1cb_3]$. Then ac meets df in an event e such that $[aec]$ and $[def]$* (see Figure 12).

A.4.1 Case (a)

Proof Case (a): If fd meets b_1b_3 in an event g such that $[b_1b_3g]$ (Figure 13a), then the Second Collinearity Theorem (Th.7) applied to the kinematic triangle $\triangle acb_3$ with $[(b_1)cb_3g]$ and $[b_3da]$ implies the existence of an event e such that $[aec]$ and $[gde]$, and the same theorem applied to the kinematic triangle $\triangle b_2b_1b_3$ with $[b_1b_3g]$

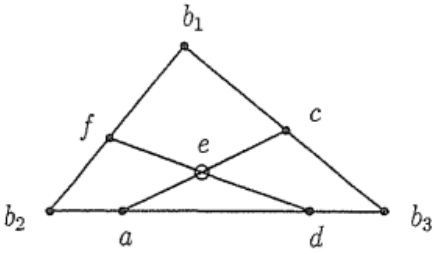

Figure 12

and $[b_3db_2]$ implies that $[gdf]$.

Theorem 8 implies that e and f are distinct so either (α) $[gdef]$ or (β) $[gdfe]$: we will show that (α) is true by demonstrating that (β) leads to a contradiction. By part (i), $[dfe]$ implies that the interval $|b_2f|$ $(\subseteq |b_2b_1|)$ meets the interval $|ae|$ $(\subseteq ac)$ in an event i such that $[aiec]$. Part (i) applied to the configuration $egcb_1f$ (which has $[efg]$ and $[gcb_1]$) implies that the segments (ec) and (b_1f) meet at an event j, but these segments belong to the paths ac and b_2b_1 respectively, so the paths meet at the two distinct events i and j: this contradicts the Axiom of Uniqueness (Axiom I3). We have now shown for Case (a) that $[aec]$ and $[def]$.

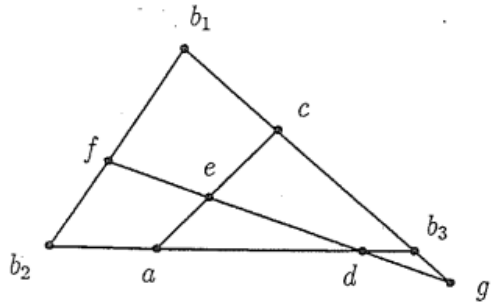

Figure 13a

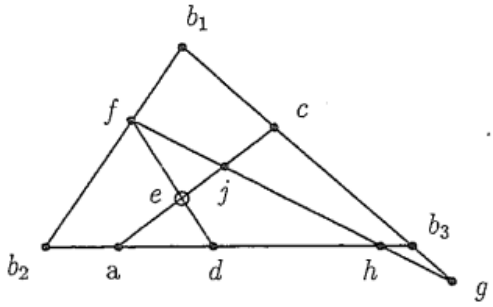

Figure 13b

A.4.2 Case (b)

Case (b): If fd does not meet b_1b_3 in an event g such that $[b_1b_3g]$ (Figure 13b), then the Second Existence Theorem (Th.14) implies that there is some event g and a path fg such that $[b_1b_3g]$. Part (i) above implies that fg meets b_2b_3 in an event h such that $[fhg]$ and $[b_2hb_3]$. Now for this case, h and d are distinct so either (α) $[b_2dhb_3]$ or (β) $[b_2hdb_3]$, but (β) leads to a contradiction since the Second Collinearity Theorem applied to the kinematic triangle $\triangle b_3gh$ with $[ghf]$ and $[hdb_3]$ would imply that fd meets b_1b_3 in an event g' (such that $[(b_1)\, b_3\, g'g]$). Thus (α) applies, so $[b_2dhb_3]$ and by Case (a), ac meets fh at an event j such that $[ajc]$, and $[hjf]$ so by part (i) fd meets ac $(= aj)$ at an event e such that $[aej(c)]$ and $[def]$. This completes the proof of Case (b) and hence the proof of Lemma 2.

A.5 Proof of Parts (ii) and (iii)

The proof of part (ii) of Theorem 15 is now complete because all possible configurations occur as special cases of part (i) and Lemma 2.

Part (iii) is an immediate consequence of part (ii), the Second Collinearity Theorem (Th.7) and Theorem 8. *q.e.d.*

Appendix B

Formalizations of Theorems and Definitions in Isabelle

B.1 Comparison of Notations

Table B.1: Comparison of Schutz's notation with the old notation in Isabelle and the new notation which does not clash with lists.

Predicate for	Schutz's notation	Old notation	New notation
Betweenness relation	$[abc]$	`[[a b c]]`	`[a;b;c]`
Non-strict betweenness	$[abcd]\!]$	`[[a b c d]]`	`[a;b;c;d]]`
Non-strict betweenness	$[\![abcd]\!]$	`[[a b c d]]`	`[[a;b;c;d]]`
Finite chain	$[a\cdots b\cdots c]$	`[f[a..b..c]X]`	`[f⇝X|a..b..c]`
Path exists & named	ab	`path a b ab`	`[a;b‖ab]`
Path exists	ab	`path_ex a b`	`[a;b]`

B.2 Mechanized Statements of Lemmas and Theorems

B.2.1 Definition of Crossing Over a Triangle

The full inductive definition is

```
inductive cross_over_tri :: ... where
two_corners:
  "⟦ℬ{a b c}∩Q={a,x}; ℬ{a b c}∩R={c,y}; [b;x;c]; [a;y;b]...⟧
  ⟹ cross_over_tri a b c Q R" |
| one_corner1:
  "⟦ℬ{a b c}∩Q={a,x}; ℬ{a b c}∩R={r,s}; [a;s;c]; [b;r;x;c]...⟧
  ⟹ cross_over_tri a b c Q R" |
| one_corner2:
  "⟦ℬ{a b c}∩Q={a,x}; ℬ{a b c}∩R={r, s};
   [a;s;c]; [a;r;b]; [b;x;c]...⟧⟹cross_over_tri a b c Q R"
| two_edges1:
  "⟦ℬ{a b c}∩Q={p,q}; ℬ{a b c}∩R={r,s};
   [a;p;r;b]; [b;q;c]; [c;s;a]...⟧⟹cross_over_tri a b c Q R"
| two_edges2:
  "⟦ℬ{a b c}∩Q={p,q}; ℬ{a b c}∩R={r,s};
   [a;p;s;c]; [b;r;q;c]...⟧⟹cross_over_tri a b c Q R"
| cross_over_tri_sym:
  "⟦cross_over_tri a b c Q R; permut3 a b c a' b' c'⟧
  ⟹ cross_over_tri a' b' c' R Q"
```

B.2.2 Crosses Triangle

First, the possible permuted cases are found:

```
lemma crosses_tri_cases:
assumes "crosses_triangle a b c P"
(* The first case implies interval x y = {x, y}. Can we
   exclude this? *)
shows "∃a' b' c'. permut3 a b c a' b' c' ∧
        ((ℬ{a' b' c'} ∩ P = {a', b'}) ∨
        (∃x. [b';x;c'] ∧ ℬ{a' b' c'}∩P={a',x}) ∨
        (∃x y. [a';x;b'] ∧ [c';y;a'] ∧ ℬ{a' b' c'}∩P={x,y}))"
```

Equivalently, we can list all cases without permuting the corners:

```
lemma crosses_tri_cases_no_perm:
assumes "crosses_triangle a b c P"
shows "ℬ{a b c}∩P={a,b} ∨ ℬ{a b c}∩P={a,c} ∨ ℬ{a b c}∩P={b,c}
    ∨ (∃x. [a;x;b] ∧ ℬ{a b c}∩P={c,x}
          ∨ [b;x;c] ∧ ℬ{a b c}∩P={a,x}
          ∨ [c;x;a] ∧ ℬ{a b c}∩P={b,x}) ∨
       (∃x y. ℬ{a b c}∩P={x,y} ∧ (
            ([a;x;b] ∧ [b;y;c])
```

```
          ∨ ([a;x;b] ∧ [c;y;a])
          ∨ ([b;x;c] ∧ [c;y;a]))"
```

Using this, several WLOG lemmas are established:

```
lemma crosses_tri_wlog:
assumes sym: "⋀a b c a' b' c'. ⟦Prop a b c P; permut3 a b c a' b' c'⟧ ⟹ Prop a' b' c' P"
    and aligned_case: "⋀a b c. ⟦△ a b c; P∈𝒫; ℬ{a b c} ∩ P = {a, b}⟧ ⟹ Prop a b c P"
    and corner_case: "⋀a b c x. ⟦△ a b c; P∈𝒫; [b;x;c]; ℬ{a b c} ∩ P = {a, x}⟧ ⟹ Prop a b c P"
    and edges_case: "⋀a b c x y. ⟦△ a b c; P∈𝒫; [a;x;b]; [c;y;a]; ℬ{a b c} ∩ P = {x,y}⟧ ⟹ Prop a b c P"
  shows "⋀a b c. ⟦crosses_triangle a b c P⟧ ⟹ Prop a b c P"
```

```
lemma crosses_tri_wlog_asm:
assumes sym: "⋀a b c a' b' c' P. ⟦Prop a b c P; permut3 a b c a' b' c'⟧ ⟹ Prop a' b' c' P"
    and symA: "⋀a b c a' b' c' P. ⟦Asm a b c P; permut3 a b c a' b' c'⟧ ⟹ Asm a' b' c' P"
    and aligned_case: "⋀a b c P. ⟦Asm a b c P; △ a b c; P∈𝒫; ℬ{a b c} ∩ P = {a, b}⟧ ⟹ Prop a b c P"
    and corner_case: "⋀a b c P x. ⟦Asm a b c P; △ a b c; P∈𝒫; [b;x;c]; ℬ{a b c} ∩ P = {a, x}⟧ ⟹ Prop a b c P"
    and edges_case: "⋀a b c P x y. ⟦Asm a b c P; △ a b c; P∈𝒫; [a;x;b]; [c;y;a]; ℬ{a b c} ∩ P = {x,y}⟧ ⟹ Prop a b c P"
  shows "⋀a b c P. ⟦Asm a b c P; crosses_triangle a b c P⟧ ⟹ Prop a b c P"
```

```
lemma crosses_tri_no_perm_wlog:
assumes aligned: "⋀ a b c P. ⟦△ a b c; P∈𝒫; ℬ{a b c}∩P={a,b} ∨ ℬ{a b c}∩P={a,c} ∨ ℬ{a b c}∩P={b,c}⟧ ⟹ Prop a b c P"
    and corner_c: "⋀ a b c P x. ⟦△ a b c; P∈𝒫; [a;x;b]; ℬ{a b c}∩P={c,x}⟧ ⟹ Prop a b c P"
    and corner_a: "⋀ a b c P x. ⟦△ a b c; P∈𝒫; [b;x;c]; ℬ{a b c}∩P={a,x}⟧ ⟹ Prop a b c P"
    and corner_b: "⋀ a b c P x. ⟦△ a b c; P∈𝒫; [c;x;a]; ℬ{a b c}∩P={b,x}⟧ ⟹ Prop a b c P"
    and edges_b: "⋀ a b c P x y. ⟦△ a b c; P∈𝒫; [a;x;b]; [b;y;c]; ℬ{a b c}∩P={x,y}⟧ ⟹ Prop a b c P"
    and edges_a: "⋀ a b c P x y. ⟦△ a b c; P∈𝒫; [a;x;b]; [c;y;a]; ℬ{a b c}∩P={x,y}⟧ ⟹ Prop a b c P"
    and edges_c: "⋀ a b c P x y. ⟦△ a b c; P∈𝒫; [b;x;c]; [c;y;a]; ℬ{a b c}∩P={x,y}⟧ ⟹ Prop a b c P"
  shows "⋀a b c P. crosses_triangle a b c P ⟹ Prop a b c P"
```

B.2.3 Two Paths Crossing a Triangle

Using the results from the previous section, we can establish the possible cases of two paths crossing a triangle:

```
lemma two_crosses_cases:
assumes "crosses_triangle a b c Q" and "crosses_triangle a b c
    R" and "Q∩R∩ℬ{a b c}={}"
shows "cross_over_tri a b c Q R ∨ (∃a' b' c'. permut3 a b c a'
    b' c' ∧ (
        (∃p q r s. ℬ{a' b' c'}∩Q={p,q} ∧ ℬ{a' b' c'}∩R={r,s} ∧
    [a';p;r;b']] ∧ [a';q;s;c']]) ∨
        (∃p q r s. ℬ{a' b' c'}∩Q={p,q} ∧ ℬ{a' b' c'}∩R={r,s} ∧
    [a';r;p;b']] ∧ [a';s;q;c']]) ∨
        (∃p q r s. ℬ{a' b' c'}∩Q={p,q} ∧ ℬ{a' b' c'}∩R={r,s} ∧
    [a';p;b'] ∧ [b';r;c'] ∧ [a';q;s;c'])))"
```

This can again be turned into a WLOG lemma:

```
lemma (in MinkowskiSpacetime) two_crosses_wlog_asm':
assumes prop_sym: "⋀ a b c Q R a' b' c'. ⟦Prop a b c Q R;
   permut3 a b c a' b' c'⟧ ⟹ Prop a' b' c' R Q"
    and asm_sym: "⋀ a b c Q R a' b' c'. ⟦Asm a b c Q R;
   permut3 a b c a' b' c'⟧ ⟹ Asm a' b' c' R Q"
    and cross_case: "⋀ a b c Q R. ⟦Asm a b c Q R;
   cross_over_tri a b c Q R⟧ ⟹ Prop a b c Q R"
    and pass1: "⋀ a b c Q R p q r s. ⟦Asm a b c Q R; △ a b c;
   Q∈𝒫; R∈𝒫; ℬ{a b c}∩Q={p,q}; ℬ{a b c}∩R={r,s}; [a;p;r;b]]; [
   a;q;s;c]]⟧ ⟹ Prop a b c Q R"
    and pass3: "⋀ a b c Q R p q r s. ⟦Asm a b c Q R; △ a b c;
   Q∈𝒫; R∈𝒫; ℬ{a b c}∩Q={p,q}; ℬ{a b c}∩R={r,s}; [a;p;b]; [b;r
   ;c]; [a;q;s;c]⟧ ⟹ Prop a b c Q R"
  shows "⋀ a b c Q R. ⟦Asm a b c Q R; crosses_triangle a b c Q
   ; crosses_triangle a b c R; Q∩R∩ℬ{a b c}={}⟧ ⟹ Prop a b c
    Q R"
```

B.2.4 Lemma 2

Following Schutz's prose, the proof of Lemma 2 consists of a case (a) and a case (b), where case (b) is shown by obtaining further events and relations to reduce it to case (a). Therefore, lemma (a) was separated out into its own lemma:

```
lemma collinearity3_2_lemma_a:
assumes
  "△ b₁ b₂ b₃" and
  "[a;c‖ac]" and "[d;f‖df]" and
  "[b₁;f;b₂]" and "[[b₂;a;d;b₃]" and "[b₁;c;b₃]" and
  case_a: "∃g∈df. [b₁;b₃;g]"  (* implies g∈df∩b₁b₃ *)
shows "∃e ∈ ac ∩ df. [a;e;c] ∧ [d;e;f]"
```

The proof of Lemma 2 can then be summarized as follows:

```
lemma collinearity3_2_lemma:
assumes
  "△ b₁ b₂ b₃" and
  "[a;c‖ac]" and "[d;f‖df]" and
  "[b₁;f;b₂]" and "[[b₂;a;d;b₃]" and "[b₁;c;b₃]"
shows "∃e ∈ ac ∩ df. [a;e;c] ∧ [d;e;f]"
proof -
  ...
  consider
    "∃g∈df∩b₁b₃. [b₁;b₃;g]" |
    "¬(∃g∈df∩b₁b₃. [b₁;b₃;g])"
    by auto
  thus ?thesis
  proof (cases)
    case 1
    thus ?thesis
      using paths collinearity3_2_lemma_a[OF assms] by blast
  next
    case 2
    (* Follow Schutz's prose *)
    ...
    (* Apply case (a) *)
    (* Use construction to obtain final result *)
  qed
qed
```

B.2.5 Theorem 15 (iii)

Using the following lemma, which can be shown by leading all cases where either path has at least three events touching the boundary to contradictions:

```
lemma collinearity3_3_lemma:
assumes "△ a b c"
    and two_paths: "Q∈𝒫" "R∈𝒫"
    and internal: "e ∈ Q ∩ R ∩ ℐ{a b c}"
    and meet_dist4: "card ((Q ∪ R) ∩ ℬ{a b c}) = 4"
  shows "crosses_triangle a b c Q ∧ crosses_triangle a b c R"
```

The final proof of Schutz's version of Theorem 15 (iii) is straightforward:

```
theorem (*15*) collinearity3_3':
  assumes "△ a b c"
      and two_paths: "Q∈𝒫" "R∈𝒫"
      and internal: "e ∈ Q ∩ R ∩ ℐ{a b c}"
      and disj: "card ((Q ∪ R) ∩ ℬ{a b c}) = 4"
    shows "cross_over_tri a b c Q R"
proof -
  have meet_int: "e ∈ Q ∩ R" "internal_event a b c e"
    using internal by auto
  have cross: "crosses_triangle a b c Q" "crosses_triangle a b
    c R"
    using collinearity3_3_lemma[OF assms] by auto
  have paths_dist: "Q ≠ R"
  proof
    assume "Q = R"
    have "card (Q ∩ ℬ{a b c}) = 2"
      using cross(1) by (metis crosses_triangle' inf_commute)
    hence "card ((Q ∪ R) ∩ ℬ{a b c}) = 2"
      using ⟨Q = R⟩ by auto
    thus False
      using disj by linarith
  qed
  show ?thesis
    using collinearity3_3[OF cross paths_dist meet_int] by
   simp
qed
```

B.3 Definition of Chains

In the following sections, the definitions for the different kind of sequence orderings are given and their subtleties briefly discussed.

B.3.1 Weak Ordering

The mechanization of definition 1 in Isabelle is slightly more involved, since we need to allow N to be a natural number *or* infinity. Allowing an argument to be infinity or natural number can achieved using Isabelle's `option` type, where a variable of type `'b option` can either be `Some b` where `b :: 'b`, or `None`. In our case, `None` will be treated as meaning infinity (i.e. no bound on the sequence). We first define the following abbreviation, where `the (Some b) = b`, which is true when either `a < b` or `b = None`:

```
abbreviation oless :: "nat ⇒ nat option ⇒ bool" where
  "oless a b ≡ b = None ∨ a < the b"
```

We can then define `weak_ordering` as follows:

```
definition weak_ordering ::
  "(nat⇒'a) ⇒
   ('a⇒'a⇒'a⇒bool) ⇒
   'a set ⇒ nat option ⇒ bool"
where "weak_ordering f ord X oN ≡
  (∀n. oless n oN ⟶ f n ∈ X) ∧
  (∀x∈X. ∃n. oless n oN ∧ f n = x) ∧
  (∀n. oless (Suc (Suc n)) oN
        ⟶ ord (f n) (f (Suc n)) (f (Suc (Suc n))))"
```

B.3.2 Local Ordering

The main difficulty in formalizing definition 2 in Isabelle is that one has to carefully consider the case of the set `X` being infinite. We cannot simply use `i < card X`, since for infinite set it is `card X = 0`. A concise definition can still be obtained, without making explicit case distinctions, by using the fact that `False ⟶ i < card X` is true. Thus, instead of something like

$$\forall i<|X|.\ \ldots,$$

using the mathematical notation where we assume the cardinality is valued infinite, we can use

$$\forall i.\ (\text{finite } X \longrightarrow n < \text{card } X) \longrightarrow \ldots.$$

Using this, local ordered chains can be implemented in Isabelle as follows:

```
definition local_ordering ::
  "(nat⇒'a) ⇒ ('a⇒'a⇒'a⇒bool) ⇒ 'a set ⇒ bool"
where "local_ordering f ord X ≡
```

```
  (∀n. (finite X ⟶ n < card X) ⟶ f n ∈ X) ∧
  (∀x∈X. (∃n. (finite X ⟶ n < card X) ∧ f n = x)) ∧
  (∀n. (finite X ⟶ Suc (Suc n) < card X)
        ⟶ ord (f n) (f (Suc n)) (f (Suc (Suc n))))"
```

B.3.3 Total Ordering

The mechanized formalization of total ordering of definition 3 is a straight-forward generalization of the mechanized formalization of local ordering:

```
definition ordering ::
  "(nat⇒'a) ⇒ ('a⇒'a⇒'a⇒bool) ⇒ 'a set ⇒ bool"
where "ordering f ord X ≡
  (∀n. (finite X ⟶ n < card X) ⟶ f n ∈ X) ∧
  (∀x∈X. (∃n. (finite X ⟶ n < card X) ∧ f n = x)) ∧
  (∀n n' n''.
      (finite X ⟶ n'' < card X) ∧ n < n' ∧ n' < n''
        ⟶ ord (f n) (f n') (f n''))"
```

B.3.4 New chain definitions

We follow Schutz's case distinction and introduce a short chain of two elements:

```
definition short_ch :: "(nat ⇒ 'a) ⇒ 'a set ⇒ bool"
  where "short_ch f X ≡ X = {f 0, f 1} ∧ [f 0; f 1]"
```

It is easy to show (as was done in Isabelle), that short chains are effectively equivalent to `path_ex`, i.e. `[a;b]`. We have `short_ch f X ⟹ [f 0; f 1]` and in the other direction `[a;b] ⟹ ∃ f. short_ch {a,b}`[1]. The above definition with an index function was chosen to be consistent with that for long chains:

```
definition long_ch :: "(nat ⇒ 'a) ⇒ 'a set ⇒ bool"
where "long_ch f X ≡
  (finite X ⟶ card X > 2) ∧ local_ordering f between X"
```

The base definition of chains is then simply

```
definition chain :: "(nat ⇒ 'a) ⇒ 'a set ⇒ bool"
where "[f↝X] ≡ short_ch f X ∨ long_ch f X"
```

The notation `f↝X` is meant to evoke *f indexes into X*.

It is now straightforward to introduce further variations of chains with several elements specified, such as `[f↝X|a..b..c]` and `[a..b..c]` (the latter is equivalent to `∃f X. [f↝X|a..b..c]`). For illustrative purposes, one such definition is given:

[1] Namely `f=(λn::nat. if x=0 then a else b)`.

```
abbreviation fin_ch ::
  "(nat⇒'a) ⇒ 'a set ⇒ 'a ⇒ 'a ⇒ bool"
where "[f⇝Q|x..z] ≡
  [f⇝Q] ∧ finite Q ∧ f 0 = x ∧ f (card Q - 1) = z"
```

Finally, a definition that avoids the case splitting into short and long chains was found and proven to be equivalent:

```
lemma chain_alt:
  "[f⇝X] ⟷ local_ordering f between X
            ∧ (infinite X ∨ card X ≥ 2) ∧ [f 0;f 1]
```