

Isabelle Formalisation of Original Representation Theorems

Marco B. Caminati^[0000-0002-4529-5442]

School of Computing and Communications
Lancaster University in Leipzig
Nikolaistrasse 10
04109 Leipzig
Germany
`m.caminati@lancaster.ac.uk`

Abstract. In a recent paper, new theorems linking apparently unrelated mathematical objects (event structures from concurrency theory and full graphs arising in computational biology) were discovered by cross-site data mining on huge databases, and building on existing Isabelle-verified event structures enumeration algorithms. Given the origin and newness of such theorems, their formal verification is particularly desirable. This paper presents such a verification via Isabelle/HOL definitions and theorems, and exposes the technical challenges found in the process. The introduced formalisation completes the verification of Isabelle-verified event structure enumeration algorithms into a fully verified framework to link event structures to full graphs.

1 Introduction

In [4], the first machine-verified contribution to the *Online Encyclopedia of Integer Sequences (OEIS)* [22] was presented, through an Isabelle/HOL-verified algorithm enumerating all labeled *prime event structures* (or just event structures, or even only *ES's*). In [7], a mining technique over massive sets of documents permitted to unearth unforeseen connections between apparently unrelated mathematical domains. One particular connection was, in the same paper, explored, linking event structures (via the algorithm from [4]) to *full graphs (FGs)*. Event structures are originated in the study of concurrent computational systems, while full graphs arise in the field of computational biology [12]. In [7], the deeper motivation of this connection was found as being given rise by a new representation theorem for event structures and a set of derived results, cross-fertilising between the two fields and permitting to obtain new theorems for both the related objects (ES's and FGs). The two papers [4] and [7], therefore, complement each other to provide enumerating algorithms and new connections found using the former. However, only the results from [4] have been mechanically checked. The present paper completes the work by providing a Isabelle/HOL (from now on, just Isabelle) [18] formalisation of the representation theorem, the theorem connecting ES's and FGs, a number of related Isabelle definitions and tools, and

a computable Isabelle isomorphism providing the connection between ES's and FGs.

Section 2 introduces the subjects of the discourse (e.g., event structures and full graphs), Section 3 provides the pen-and-paper version of the theorems formalised, Section 4 illustrates the main formalised theorems and definitions, Sections 5 and 6, respectively, illustrate the formalisation of the two main theorems, while Section 7 contains overall consideration about the formalisation process. Section 8 concludes.

2 Event Structures and Full Graphs

This section formally introduces the objects of our theorems. To make this paper self-contained, it summarises, together with the subsequent one, the main elements of Sections IV and VI of [7].

2.1 Event Structures

A prime event structure (or simply event structure, ES) describes a concurrent computation by identifying the computational events that are causally related and those that exclude one another. According to the following definition, this is achieved via two relations: \leq (causality) and $\#$ (conflict).

Definition 1. *An event structure is a pair of relations $(\leq, \#)$ where \leq is a partial order, $\#$ is irreflexive and symmetric, $(\text{fie } \leq) \supseteq (\text{fie } \#)$ is called the set of events, and for any three events x_0, x_1, y : $x_0 \# y \wedge x_0 \leq x_1 \rightarrow x_1 \# y$.*

The last condition is referred to as conflict propagation. In Definition 1, fie denotes the field of a relation: that is, the union of its domain (dom) and range (ran). The usual infix notation for the relations in Definition 1 can become inconvenient, therefore we also introduce an additional notation representing the relations with letters, writing, e.g., $(x, y) \in D$ instead of $x \leq y$ and $(x, y) \in U$ in lieu of $x \# y$. We will typically use the letters D and U as above to suggest the reader what they encode: D stands for “directed” and U for “undirected”. Indeed, \leq , as a partial order, is naturally viewable as a directed graph and $\#$, being symmetric, as an undirected graph. See also the comment immediately after Definition 2. Since any finite relation is a graph having its vertices (or nodes) coinciding with the field of the relation, and since, for any finite partial order, that graph can be naturally made a directed graph, it is easy to represent any finite ES via diagrams such as the one in Fig. 1.

2.2 Full Graphs

Any family of sets can be used to build a graph where each vertex represents a set of the family, an undirected edge connects overlapping sets, and a directed edge connects a superset to a subset. Such a construction occurs when studying

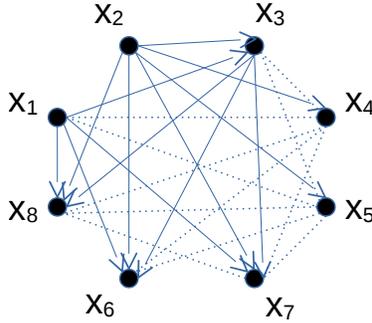


Fig. 1. An example event structure, with eight events related by causality (denoted by an arrow standing for \leq) and conflict (denoted by a dashed line).

the following problem: given n subsets of a given set of m elements, is there a way of labeling the elements with natural numbers such that the element occur consecutively (with respect to this labeling) in each subset? One practical application of this labeling problem arises in bioinformatics, where the elements of subsets represent observed blemishes to parts of a gene, which are supposed to be more likely to affect parts of the gene which are connected: therefore, finding such a labeling can provide essential information about the topology of a gene [2, 12]. The graphs that can be created in this way are specified by Definition 2.

Definition 2. A full graph (FG) is a mixed, unweighted, simple¹ graph over vertices V , of directed edges D , and undirected edges T such that there is an injective function f on V yielding non-empty sets and with the property

$$\forall x, y \in V. ((x, y) \in D \leftrightarrow f(x) \supseteq f(y)) \wedge \quad (1)$$

$$((x, y) \in T \leftrightarrow f(x) \text{ and } f(y) \text{ overlap}); \quad (2)$$

here, we say that two sets A and B overlap (written $A \checkmark B$) when $A \cap B \notin \{A, B, \emptyset\}$. We call f an fg-representation of the full graph (D, T) . Alternatively, we will say that T makes a full graph of D (through f) when such an fg-representation f exists.

Having insisted in Definition 2 in encoding T via ordered pairs, even though is an undirected graph, makes that encoding redundant; however, this is convenient because we can then regard T as a (symmetric) relation, as all the other components in the definitions of ES's and FGs, also thanks to the fact that all these components are simple graphs, making the encoding as relations adequate.

¹ Recall that a graph is *simple* when it has no self loops and no multi-edges; it is *mixed* when it has both directed and undirected edges. See [13, Section 1.1].

3 Connecting ES's and FGs

In [7], a systematic way of looking for matches between entries in the OEIS and free text search results across Google and Google Scholar is introduced, producing thousands of unexplored and potentially interesting matches. One of them relates the enumerations of ES's (introduced in OEIS by [4]) and of FGs (in [9, Section 4]): the number of (labeled) ES's and of FGs over a fixed number of vertices n coincide for small n . In the same paper [7], this connection is explored, motivated and proven by providing a one-to-one map between ES's and FGs, which is an isomorphism once framed as a mapping between representations of ES's and FGs.

To introduce the ideas in the latter paper, we start by looking at the evaluation, through this isomorphism, of the ES in Fig. 1, giving the FG in Fig. 2.

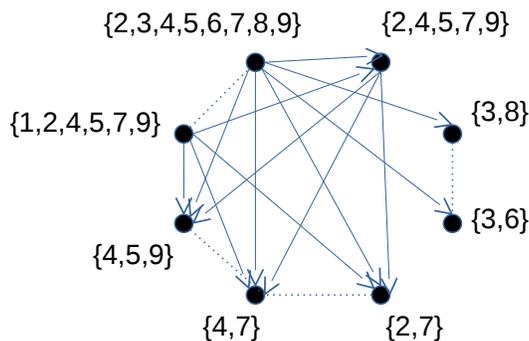


Fig. 2. The full graph isomorphic to the event structure of Fig. 1. This is the full graph example originally featured in Section 3 of [12]. Here, the arrows represent \supseteq , and the dashed lines the overlapping relation.

To make more precise the similarity between the figures, we must understand how they are generated: in Fig. 2, the edges are determined by looking at operations on sets associated to each node. In this sense, we have a representation of the FG in terms of set-theoretical notions, indirectly dictating the structure of the FG itself by definition. In the case of event structures, however, such a representation is absent in the definition, which dictates the property of the structure directly by imposing relationships between \leq and $\#$. To formally link the connection we are looking at, we must find a representation for the ES as well, through a suitable definition of ES-representation and a *representation theorem* establishing that an equivalent definition of ES can be given in terms of such a representation, as done with FGs. This is an interesting endeavour in general, not limited to the specific task of finding connections between different domains: see [7, Section III], which also discusses and details the notion of representation.

The following definition will turn out to yield adequate representations for ES's.

Definition 3. *Given two binary relations D and U , the set-valued function f is a representation for (D, U) if*

$$\forall x, y \in \text{dom } f. ((x, y) \in D \leftrightarrow f(x) \supseteq f(y)) \wedge \quad (3)$$

$$\forall x, y \in \text{dom } f. ((x, y) \in U \leftrightarrow f(x) \cap f(y) = \emptyset). \quad (4)$$

Here, we say that, given D and any U with $\text{fie } U \subseteq \text{fie } D$, any such a f (if it exists) is called *admissible*.

And by adequate we mean that the following representation theorem holds.

Theorem 1 (Representation theorem). *Consider two binary relations D and U , with D finite and $\text{fie } U \subseteq \text{fie } D$. Then (D, U) is an event structure if and only if there is an injective representation $f : \text{fie } D \rightarrow \overline{2}^{\mathbb{N}} \setminus \{\emptyset\}$ for (D, U) ,*

where $\overline{2}^X$ denotes the finite subsets of X .

Our second representation theorem for event structures, Theorem 2, offers a bijective construction connecting them to full graphs.

Theorem 2. *Consider a finite relation D and a function F_D mapping working as follows on its argument R :*

$$F_D := R \mapsto (\text{fie } D \times \text{fie } D) \setminus (D \cup D^{-1}) \setminus R.$$

A bijection between

$X := \{T \mid T \text{ makes a full graph of } D\}$ *and*

$Y := \{U \mid U \text{ is admissible for } D\}$

is given by $F_D|_X$.

Fig. 3 attaches a representation (always existing, according to Theorem 1) to the ES of Fig. 1. Using F_D as in Theorem 3, one can now promptly relate that ES to the FG of Fig. 2.

4 Formalisation and Verification: Introduction

We start from the top level, that is, the Isabelle renditions of the main theorems.

Theorem 1 is stated as

Listing 1.1. Isabelle rendition of Theorem 1

```
theorem representation: assumes "finite D"
  "Field U ⊆ Field D" shows
  "(isLes D U) = (∃ f. isInjection f & Domain f = Field D &
    ({::nat set} ⊄ Range f & finite ((Union o Range) f)
    & isRepresentation f D U)",
```

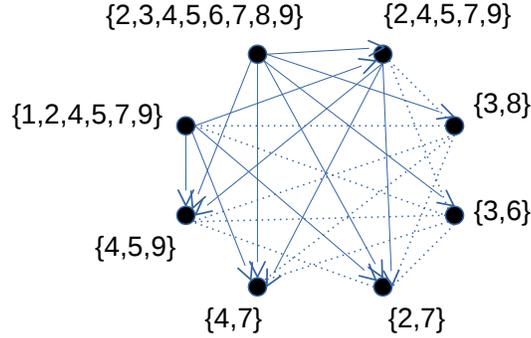


Fig. 3. A representation for the event structure of Fig. 1. Now, the arrows represent \supseteq and the dashed lines the disjointness relation. Theorem 1 states that any set of events is an event structure if and only if such a representation is constructible.

while Theorem 2 reads

Listing 1.2. Isabelle rendition of Theorem 2

```

theorem bijection: assumes "finite D"
  "F=(λR. ((Field D × Field D) - (D ∪ D-1)- R))"
  "X={T|T. Field T ⊆ Field D & (∃ f. isInjection f &
    ({::nat set)∉Range f & Domain f=Field D &
    isFgRepr f D T})"
  "Y={U|U. Field U ⊆ Field D & (∃ f. isInjection f &
    ({::nat set)∉Range f & Domain f=Field D &
    isRepresentation f D U})" shows
  "F'X=Y & F'Y=X & inj_on F X & inj_on F Y & card X=card Y",

```

where $\text{inj_on } F \ X$ returns true when the total function F is injective over the set X , while the notation \sim^{-1} denotes the converse of a relation.

The reader might have noticed a subtle difference between f occurring in Listing 1.1 and F occurring in Listing 1.2: while both are functions, they are implemented very differently within Isabelle/HOL. Indeed, F is a standard HOL function, a primitive notion in higher order logic [16]; on the other hand, f is implemented as a set of ordered pairs, in the way standard set theory (e.g., ZF, Zermelo-Fraenkel set theory [11]) represents functions. The verification presented here extensively exploits this duality, choosing one construct or the other depending on the particular function at hand and on the theorem it appears in. There are several reasons for this approach: one is that the totality of functions imposed by HOL is sometimes an inconvenience [16] which can be worked around by choosing the second construct; another one is that set theoretical operations on functions, such as union, subtraction, conversion (\sim^{-1}) are sometimes useful, and are unavailable with the first construct; as an example of this usefulness, let us take the \leftarrow infix operator, which grows a relation P with another one Q , performing overriding if necessary, and is defined as

$(P - (\text{Domain } Q \times \text{Range } P)) \cup Q$.

One advantage of this definition is that it works for any pair of relations P and Q , and at the same time preserves right-uniqueness if P and Q are right-unique (that is, functions). Additionally, existing facts about the building blocks of $+<$ ($-$, \times , Domain , Range , \cup) typically makes proofs about $+<$ easier, helped by the simplicity of its definition. This operator can be conveniently overloaded to the point-wise special case:

```
abbreviation singlepaste where "singlepaste f pair ==
f +< {(fst pair, snd pair)}"
notation singlepaste (infix "+<" 75)
```

Note that the type of g in $f+<g$ avoids ambiguity for the overloaded $+<$ operator.

On the other hand, set-theoretical functions are actually relations and, as such, need to be shown to be right unique (by showing they satisfy a dedicated Isabelle predicate `runiq`) before they can be treated as functions. Overall, keeping both constructs has the upside of being able to take advantage of the best of both worlds [8].

The price to pay for this upside is that we have duplicated versions of most operations on functions, one for each construct. For example, if F is a standard HOL function and f is a set theoretical function, then the application operation on an argument x is written $F x$ for F and $f ,, x$ for f ; the operation yielding the image of a set X through the function is $F'X$ versus $f'X$, the range operation is `range F` versus `Range f`; the property of injectivity is `inj_on` versus `isInjection`, etc. Other operations, such as union, intersection, domain, \sim^{-1} , restriction (denoted `||`), and others, only make sense for set-theoretical functions, although a restriction operating on HOL functions (and denoted `|||`) was also introduced. In this case, naturally, the result is a set-theoretical functions, since in HOL all functions are total and cannot therefore be restricted directly [16].

In practice, the reader needs not to worry about these subtle differences deriving from the duality between HOL functions and set-theoretical functions, which were nevertheless discussed in the digression above to prevent confusion.

The first theorem above, in Listing 1.1, equates the definition of being an event structure (`isLes`) to the existence of a representation (whose definition is contained in `isRepresentation`), while the second theorem shows that F (the Isabelle rendition of F_D occurring in Theorem 2) is indeed a bijection between the set Y of admissible conflicts for D and the set X of undirected graphs making D a full graph. Since this holds for all finite D s, we have a verified proof of the mined matches illustrated in Section 1 and in [7].

`isLes`, `isRepresentation`, `isFgRepr` are all straightforward from the pen-and-paper definitions, with the first already used in previous formalisations regarding event structures [4–6]:

```
definition "isLes causality conflict =
propagation conflict causality & sym conflict &
irrefl conflict & trans causality &
antisym causality & reflex causality",
```

```

definition "isRepresentation f D U =  $\forall x \in \text{Domain } f.$ 
  ( $\forall y \in \text{Domain } f. ((x, y) \in D) = (f, x \supseteq f, y)$ ) &
  ( $((x, y) \in U) = ((f, x \cap f, y) = \{\})$ )"

```

```

definition "isFgRepr f D T =  $\forall x \in \text{Domain } f.$ 
  ( $\forall y \in \text{Domain } f. ((x, y) \in D) = (f, x \supseteq f, y)$ ) &
  ( $((x, y) \in T) = ((f, x) \text{ overlaps } (f, y))$ )",

```

with the definition of overlapping also very close to the paper version and taking advantage of the infix notation definition capabilities of Isabelle:

```

definition "Overlap X Y =  $(X \cap Y \notin \{X, Y, \{\}\})$ "
notation "Overlap" ("_ overlaps ")

```

Moreover, `propagation` is a synonym for the following:

```

definition "isMonotonicOver conflict causality =
 $\forall x y. (x, y) \in \text{causality} \rightarrow \text{conflict } \{x\} \subseteq \text{conflict } \{y\}$ ",

```

while `reflex` was introduced as follows:

```

definition "reflex P = refl_on (Field P) P",

```

where `refl_on A R` returns true when the relation `R` is reflexive over a subset `A` of its domain and range.

All the other Isabelle objects occurring above are part of Isabelle's standard library.

5 Formalisation and Verification: Proof Structure for bijection

We start from the second theorem introduced above, which is the simpler of the two, in that it relates full graphs to sets of admissible conflict relations for a given partial order, while the link between ES representations and ES's is provided by `representation`.

The idea for the proof is simple: we just note that the definition of fg-representation (Definition 2) and of event structure representation (Definition 3) are very similar, mainly differing by the substitution of the overlapping relation with that of disjointness; therefore, we introduce the following operator to map between them:

```

 $\lambda R. (\text{unRel}' D - R),$ 

```

where the helper `unRel'` takes the complement of a relation:

```

abbreviation "unRel' D == (Field D  $\times$  Field D) - (D  $\cup$  D-1)".

```

Now, the idea is to show that we can pass from event structures to full graphs by applying the above operator to the conflict relation. To show that, it suffices to show that the set of valid undirected edges for a given `D` can be obtained from the set of valid conflict relations for `D` by applying the operator above: this is exactly the thesis $F'X=Y \ \& \ F'Y=X$ appearing in the `bijection` theorem's thesis.

By bijectivity, it suffices to show the weaker relations $F'X \subseteq F'Y$ and $F'Y \subseteq F'X$, which is done by 153a and 153b below, respectively:

```
lemma 153a: assumes "F=(λR. (unRel' D - R))" shows
"F' {T|T. Field T ⊆ Field D & (∃ f. isInjection f
& ({}::nat set)∉Range f & Domain f=Field D
& isFgRepr f D T)} ⊆
{U|U. Field U ⊆ Field D & (∃ f. isInjection f &
({}::nat set)∉Range f & Domain f=Field D
& isRepresentation f D U)}"
```

```
lemma 153b: assumes "F=(λR. (unRel' D - R))" shows
"F' {U|U. Field U ⊆ Field D & (∃ f. isInjection f &
({}::nat set)∉Range f & Domain f=Field D &
isRepresentation f D U)} ⊆
{T|T. Field T ⊆ Field D & (∃ f. isInjection f &
({}::nat set)∉Range f & Domain f=Field D &
isFgRepr f D T)}"
```

153a and 153b are sufficient to draw the thesis of `bijection` thanks to the following general propositions (the latter provided by Isabelle's standard library):

```
proposition 152a: assumes "finite (X ∪ Y)" "inj_on f X"
"inj_on f Y" "f'X ⊆ Y" "f'Y ⊆ X" shows "f'X=Y & f'Y=X"
```

```
lemma card_image:
assumes "inj_on f A"
shows "card (f ' A) = card A"
```

Finally, the hypotheses `inj_on f X` and `inj_on f Y` can be deduced when `X` and `Y` are, respectively, the sets appearing in 153b by another general result:

```
proposition 155: "inj_on (λX. Y-X) (Pow Y)"
```

(where `Pow` takes the power set), which applies when `X` and `Y` take the particular values above thanks to

```
lemma 154bb: assumes "isFgRepr f D T" "Domain f = Field D"
"Field T ⊆ Field D" shows "T ⊆ (Field D × Field D)-(D∪D-1)"
```

and

```
lemma 154aa: assumes "isRepresentation f D U"
"({}::nat set)∉Range f" "runiq f" "Domain f = Field D"
"Field U ⊆ Field D" shows "U ⊆ (Field D × Field D)-(D∪D-1)",
```

where the `runiq` predicate was introduced in the discussion after Listing 1.2.

6 Formalisation and Verification: Proof Structure for representation

The proof is in the two directions; that is, having a representation implies being an event structure (theorem `main1`) and being an event structure implies having a representation (theorem `main2`):

```

theorem main1: assumes "runiq f"
"Field D  $\cup$  Field U  $\subseteq$  Domain f"
"isRepresentation' f D U"
  shows
"isPreorder D & isMonotonicOver U D & sym U &
(luniq f  $\rightarrow$  antisym D) & ( $\{\}$  $\notin$ (Range f)  $\rightarrow$  irrefl U)"

theorem main2: assumes "finite D" "isLes D U" obtains
f::('a  $\times$  nat set)set" where "Domain f=Field D &
  isInjection f &  $\{\}$  $\notin$ Range f &
  finite ((Union o Range) f) & isRepresentation f D U"

```

6.1 Proof of main2

The proof for main2 is arguably among the most complex in the project, since it needs to provide a representation for any given ES. It is done by induction on the cardinality of D, starting with the base case which can be proved by Sledgehammer [3]:

```

proposition 1150a: assumes "f= $\{\}$ " "D= $\{\}$ " shows
"isRepresentation f D U & Domain f=Field D &
isInjection f & runiq f &  $\{\}$  $\notin$ Range f"

```

The induction step now requires to somehow pass from a representation f of a D' smaller than a given D to a representation for D itself. This requires to determine two things:

1. in which sense D' is smaller than D ;
2. how to construct the new representation from f .

For (1), we set D' and D to differ by exactly one *terminal* event: that is, D' is obtained from D by removing one event s with no children in D .

For (2), we obtain the new representation for D by just growing f with one new set RA representing s ; this growth is done by the $+<$ operator seen in Section 4. Note that this growth does not affect the values f has on the old events. Theorem `extension2` below does exactly that, showing that the function resulting from the $+<$ operation is still a representation for D . However, for this thesis to hold, there are three fundamental requirements on RA , the set representing the new event s ; these requirements must hold for any existing event x , and appear in the hypotheses of `extension2` labeled as `hypOverlap`, `hypCausality` and `hypConflict`. The remaining hypotheses are merely technical, expressing obvious requirements such as f needing to be a function, s having no children, s being fresh, etc.

```

theorem extension2: assumes "runiq f" "(s,s) $\in$ D"
"D' '{s} $\subseteq$ {s}" "s $\notin$ Domain f" assumes
hypOverlap: " $\forall x \in$ Domain f.  $\neg(f, , , x \subseteq RA)$ " assumes
hypCausality: " $\forall x \in$ Domain f.  $RA \subseteq f, , , x = (x \in D^{-1} '{s} - \{s})$ "
assumes
hypConflict: " $\forall x \in$ Domain f.  $((f, , , x) \cap RA = \{\}) = (x \in U^{-1} '{s})$ "

```

```

"∀x∈Domain f. ((x,s)∈U) = ((s,x)∈U)"
"isRepresentation f (D---s s) (U---s s)"
"F=f+<(s,RA)" "RA≠{}" "(s,s)∉U"
  shows
"isRepresentation' F D U"

```

extension2 presents a couple of new constructs: first, the operator --- allows to remove a pair from a relation, so that, in this case, D and U are extensions of D---s s and U---s s. Secondly, the operator ,, is very similar to ,, seen in Section 4, but with a slightly more general definition which is technically more convenient in some cases. Let us start with the definition of ---:

```

definition "bouthside P X Y =
  P - ((X×Range P) ∪ ((Domain P)×Y))"
notation "bouthside" ("_\\")
definition "singlebouthside P x y = bouthside P {x} {y}"
notation "singlebouthside" ("_---")

```

This definition uses a special case of \\, which merely removes portions of domain and range from any relation using elementary set-theoretical operations.

extension2 is what we need to obtain our representation theorem. However, as we mentioned above, it dictates three conditions on RA (hypOverlap, hypCausality and hypConflict) for its validity. We therefore need to build a set RA satisfying them. The following result, one of the most technical, builds a suitable RA, by transforming the representation f occurring in extension2 into an intermediate representation g before inducting.

```

lemma l46: assumes "isRepresentation f (D---s s) (U---s s)"
"runiq f & D' '{s}={s} & sym U &
(let dm=Domain in let R=Range in {}∉R f &
finite ((Union o R) f) & (Domain D)-{s} ⊆ dm f &
  (let d=D---s s in dm f ⊆ Range d & trans d))"
"let d=D---s s in let sparents=d^-1' '(D^-1' '{s}) in
let sconfl=U^-1' '{s} in
let sconcurs=Range d-(sparents ∪ sconfl) in
  finite sconcurs & sconcurs⊆fixPts D &
  sparents=D^-1' '{s}-{s} &
  irrefl (U|(sconcurs ∪ Domain f)) &
  sconfl ∩ D^-1' '{s}={} &
  d' 'sconfl⊆sconfl &
  isMonotonicOver U (D|^(D^-1' '{s} ∪ (Range d - sconfl)))"
shows
"∃ l. let N=Max ((Union o Range) f)+1+size l in
let d=Domain in let R=Range in
let RA=(Union o set)((map (Union o R) l)@[{N}]) in
let g=foldl pointUnion f (l@[D^-1' '{s}-{s}]×[{N}]) in
let h=g+<(s,RA) in d g=d f & d h=d f ∪ {s} & {}∉R g &
{}∉R h & isRepresentation g (D---s s) (U---s s) &
isRepresentation h D U & runiq g & runiq h &
(Union o R) h ⊆ {0..<1+N} &
(luniq f → (isInjection g & isInjection h))"

```

Although harder to read than `extension2`, `146` has the advantage of having moved all the requirements on `RA` back to the given event structure (D, U) . This comes at the price of passing through `g`, which is obtained from `f` by repeatedly applying the following operator `pointUnion` to the given `f` over a suitable list of sets, through the standard functor `foldl`:

```
definition "pointUnion ff A =
  ff +< ((λx. ff , , x ∪ A , , x) ||| (Domain A))".
```

Recall that `|||` is the restriction operator, see Section 4.

6.2 Proof of `main1`

The proof of theorem `main1` is less technical, and is nicely broken into sublemmas each providing a part of the thesis. The following lemma takes care of the transitivity:

```
lemma 149a: assumes "runiq f" "Field D ⊆ Domain f"
  "∀x0∈Domain f. (∀x1∈Domain f. (((x0, x1)∈D)=(f, ,x0⊇f, ,x1)))"
  shows "Field D ⊆ fixPts D & trans D",
```

(where definition `"fixPts P=Domain(Id∩P)"`), while this other proposition takes care of conflict propagation:

```
proposition 149bb: assumes "Field D ∪ Range U ⊆ Domain f"
  "∀x∈Domain f. (∀y∈Domain f. (((x, y)∈D)→(f, ,x ⊇ f, ,y)) &
    (((x, y)∈U) = ((f, ,x ∩ f, ,y)={})))"
  shows "isMonotonicOver U D"
```

The reflexivity is then granted by combining `149a` with this simple but useful fact:

```
proposition 145e: "(∀x∈Field P. (x, x)∈P)=reflex P".
```

When writing the formalisation, a guiding principle was to always try to derive particular results from weaker results (whether the latter already exist in some library or not) applicable to more general objects, which can be strengthened to be applied to more particular objects needed in the specific formalisation one is carrying out. This resulted in over 300 lemmas, propositions and theorems, and around 50 new objects defined.

Isabelle was also used to work out minimal requirements for particular results. For example, in `main1`, no finiteness is required over `D`, and the particular irreflexivity property is explicitly bound to the additional requirement of `f` not yielding the empty set as a representation. Similarly, in theorem `main1` the anti-symmetry property of event structures is linked to the representation being an injection. These details add proof-theoretical information to any development, and are usually hard to keep track of manually with a pen-and-paper proof.

7 The Formalisation Process

The code is available at² <https://gitlab.com/users/mbc8/contributed>. The formalisation of the mathematical objects and results introduced above is roughly 2.7kSLOC and 151Kb (36Kb gzipped) of Isabelle code; a bit more due to spawned additions to the theories created for event structures for previous papers such as [4–6]. To quantitatively assess the formalisation, the length of the mathematical parts appearing in [7] was computed by converting the relevant pdf to text, obtaining 21502 bytes (8229 gzipped) as a result. This gives an apparent de Bruijn factor of 7, and an intrinsic one of 4.3. There are about 4 pages of mathematical content in [7], whereas the time spent to formalise it has been estimated in around two weeks of work, giving a formalisation cost of 0.5 weeks per page. All these numerical parameters are approximate, but help giving an idea of the process itself [1, 17]. It should also be noted that, although Isabelle/HOL implementations of graph theory abound ([15, 19–21]), the present formalisation used none of them, for two reasons: first, although the theorems relate event structures and full graphs, they don't really need much graph theory. Not even basic notions as walks, paths, etc. are even mentioned. Secondly, our formalisation deals with mixed graphs (i.e., having both directed and undirected edges), thus restricting the available libraries. The theorem `representation` uses 141 facts (including lemmas, propositions, theorems and definitions) included in the file `fullGraph.thy`. The proofs can be divided into automatically generated ones and one with an explicit Isar proof (starting with the `proof` keyword). A minority of those explicit proofs were generated by Sledgehammer's `isar_proof` feature, but most of them were manually written. In general, the preference is to have small general facts with simple, usually automatic proofs, which are then put together for the more complex, manual proofs. This yields a proliferation of lemmas which are hopefully reusable. This approach goes hand-in-hand with the one providing definitions built in blocks on top of more general definitions. For example, `pointUnion` is defined in terms of `|||` and `+<`, which are in turn defined in terms of elementary set theoretical operations (cartesian product, union, intersection, set difference, etc.). One of the longest proof is that of 146 (see Section 6.1), which is 139 lines and about 10Kb. About 10 results have proof longer than 20 lines, usually substantially longer, and a number of them has to do with the problem of suitably constructing `RA` using a reiterated (via `fold1`) `pointUnion` operation (see Section 6.1). Most proofs are non-constructive; for example, they do not provide an algorithm to build representations. However, the operator `F` appearing in theorem `bijection` and allowing to pass from representations to fg-representations and vice-versa is computable.

8 Conclusions

This paper has presented a rare instance of original theorems having been formalised natively: they were born formalised. More than that, they were discov-

² The link requires a reasonably recent browser.

ered thanks to existing formalisations. Such theorems provide new representations for event structures and unexpectedly link the latter to the unrelated field of computational biology through the notion of full graphs. This permits to apply results from one domain to another to immediately obtain new theorems (some such examples are reported in [7]) Therefore, an obvious idea for future work is to formally verify these new theorems, which would imply a formalisation for the domain which is currently not formalised: that of full graphs. Indeed, while event structures have now a reasonable amount of results formalised, no formalisation exists for more advanced results applicable to full graphs, for example those in [9, 14].

Looking at automated theorem proving, the origin of the presented results (obtained via data mining, as explained in [7]), can provide avenues to both develop new techniques and test existing ones: thousands of potentially interesting matches similar to the one giving rise to the results presented here were found.

Another future work direction will seek the generalisation of the original theorems presented here: one natural idea is the extension of Theorem 1 to infinite event structures, which is comparable to how Priestley’s representation theorem generalises (in a by no means trivial manner!) Birkhoff’s [10, Theorem 11.23].

References

1. Asperti, A., Sacerdoti Coen, C.: Some considerations on the usability of interactive provers. In: Intelligent Computer Mathematics: 10th International Conference, Aisc 2010, 17th Symposium, Calculemus 2010, and 9th International Conference, Mkm 2010, Paris, France, July 5-10, 2010. Proceedings. p. 147 (2010)
2. Benzer, S.: On the topology of the genetic fine structure. Proceedings of the National Academy of Sciences **45**(11), 1607–1620 (1959)
3. Blanchette, J.C., Böhme, S., Paulson, L.C.: Extending Sledgehammer with SMT solvers. Journal of Automated Reasoning **51**(1), 109–128 (2013)
4. Bowles, J.K., Caminati, M.B.: A verified algorithm enumerating event structures. In: Intelligent Computer Mathematics (CICM). pp. 239–254. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), Springer, Netherlands (5 2017)
5. Bowles, J.K., Caminati, M.B.: Balancing prescriptions with constraint solvers. In: Liò, P., Zuliani, P. (eds.) Automated Reasoning for Systems Biology and Medicine, pp. 243–267. Springer (2019)
6. Bowles, J.K., Caminati, M.B., Cha, S.: An integrated framework for verifying multiple care pathways. In: Eleventh International Symposium on Theoretical Aspects of Software Engineering (TASE). IEEE Computer Society, United States (5 2017)
7. Caminati, M.B., Bowles, J.K.F.: Representation theorems obtained by mining across web sources for hints. In: 6th International Conference on Information and Computer Technologies (ICICT). IEEE (2023), In press, preprint available at <https://eprints.lancs.ac.uk/id/eprint/185196/>
8. Caminati, M.B., Kerber, M., Lange, C., Rowat, C.: Set theory or higher order logic to represent auction concepts in Isabelle? In: Davenport, J.H., Watt, S.M., Sexton, A.P., Sojka, P., Urban, J. (eds.) International Conference on Intelligent Computer Mathematics. LNAI, vol. 8543, pp. 236–251. Springer (2014). <https://doi.org/10.1007/978-3-319-08434-3>

9. Cowen, L.J., Kleitman, D.J., Lasaga, F., Sussman, D.: Enumeration of full graphs: Onset of the asymptotic region. *Studies in Applied Mathematics* **96**(3), 339–350 (1996)
10. Davey, B., Priestley, H.: *Introduction to Lattices and Order*. Cambridge mathematical textbooks, Cambridge University Press (2002)
11. Enderton, H.B.: *Elements of set theory*. Academic press (1977)
12. Fulkerson, D., Gross, O.: Incidence matrices and interval graphs. *Pacific Journal of Mathematics* **15**(3), 835–855 (1965)
13. Gross, J.L., Yellen, J.: *Handbook of graph theory*. CRC press (2003)
14. Kleitman, D.J., Lasaga, F.R., Cowen, L.J.: Asymptotic enumeration of full graphs. *Journal of Graph Theory* **20**(1), 59–69 (1995)
15. Koutsoukou-Argyraiki, A., Bakšys, M., Edmonds, C.: A formalisation of the Balog–Szemerédi–Gowers theorem in Isabelle/HOL. In: *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*. pp. 225–238 (2023)
16. Müller, O., Slind, K.: Treating partiality in a logic of total functions. *The Computer Journal* **40**(10), 640–651 (1997)
17. Naumowicz, A.: An example of formalizing recent mathematical results in Mizar. *Journal of Applied Logic* **4**(4), 396–413 (2006)
18. Nipkow, T., Paulson, L.C., Wenzel, M.: *Isabelle/HOL: a proof assistant for higher-order logic*. Springer-Verlag, London, UK (2002)
19. Nordhoff, B., Lammich, P.: Dijkstra’s shortest path algorithm. *Archive of Formal Proofs* (2012)
20. Noschinski, L.: Proof pearl: A probabilistic proof for the Girth-Chromatic number theorem. In: *Interactive Theorem Proving: Third International Conference, ITP 2012, Princeton, NJ, USA, August 13–15, 2012. Proceedings 3*. pp. 393–404. Springer (2012)
21. Noschinski, L.: A graph library for Isabelle. *Mathematics in Computer Science* **9**(1), 23–39 (2015)
22. Sloane, N.J.A.: The on-line encyclopedia of integer sequences. In: *Annales Mathematicae et Informaticae*. vol. 41, pp. 219–234 (2013)