FORMALIZED MATHEMATICS
Volume 13, Number 2, Pages 235–252
University of Białystok, 2005

Alternative Graph Structures[1]

Gilbert Lee[2]
University of Victoria
Victoria, Canada

Piotr Rudnicki
University of Alberta
Edmonton, Canada

Summary. We define the notion of a graph anew without using the available Mizar structures. In our approach, we model graph structure as a finite function whose domain is a subset of natural numbers. The elements of the domain of the function play the role of selectors for accessing the components of the structure. As these selectors are first class objects, many future extensions of the new graph structure turned out to be easier to formalize in Mizar than with the traditional Mizar structures.

After introducing graph structure, we define its selectors and then conditions that the structure needs to satisfy to form a directed graph (in the spirit of [13]). For these graphs we define a collection of basic graph notions; the presentation of these notions is continued in articles [16, 15, 17].

We have tried to follow a number of graph theory books in choosing graph terminology but since the terminology is not commonly agreed upon, we had to make a number of compromises, see [14].

MML identifier: `GLIB_000`, version: `7.5.01 4.39.921`

The papers [20], [19], [22], [21], [24], [2], [1], [25], [7], [5], [12], [3], [8], [6], [23], [9], [4], [10], [11], and [18] provide the terminology and notation for this paper.

1. Definitions

A finite function is called a graph structure if:

(Def. 1) $\operatorname{dom} \mathrm{it} \subseteq \mathbb{N}$.

The natural number VertexSelector is defined as follows:

(Def. 2) VertexSelector = 1.

[1]This work has been partially supported by NSERC, Alberta Ingenuity Fund and iCORE.
[2]Part of author's MSc work.

The natural number EdgeSelector is defined as follows:

(Def. 3) EdgeSelector $= 2$.

The natural number SourceSelector is defined by:

(Def. 4) SourceSelector $= 3$.

The natural number TargetSelector is defined by:

(Def. 5) TargetSelector $= 4$.

The non empty subset the graph selectors of $\mathbb{N}$ is defined by:

(Def. 6) The graph selectors $=$ $\{$VertexSelector, EdgeSelector, SourceSelector, TargetSelector$\}$.

Let G be a graph structure. The vertices of G is defined by:

(Def. 7) The vertices of $G = G($VertexSelector$)$.

The edges of G is defined by:

(Def. 8) The edges of $G = G($EdgeSelector$)$.

The source of G is defined by:

(Def. 9) The source of $G = G($SourceSelector$)$.

The target of G is defined by:

(Def. 10) The target of $G = G($TargetSelector$)$.

Let G be a graph structure. We say that G is graph-like if and only if the conditions (Def. 11) are satisfied.

(Def. 11) VertexSelector $\in \operatorname{dom} G$ and EdgeSelector $\in \operatorname{dom} G$ and SourceSelector $\in \operatorname{dom} G$ and TargetSelector $\in \operatorname{dom} G$ and the vertices of G is a non empty set and the source of G is a function from the edges of G into the vertices of G and the target of G is a function from the edges of G into the vertices of G.

Let us note that there exists a graph structure which is graph-like.

A graph is a graph-like graph structure.

Let G be a graph. Observe that the vertices of G is non empty.

Let G be a graph. Then the source of G is a function from the edges of G into the vertices of G. Then the target of G is a function from the edges of G into the vertices of G.

Let V be a non empty set, let E be a set, and let S, T be functions from E into V. The functor createGraph(V, E, S, T) yielding a graph is defined by:

(Def. 12) createGraph$(V, E, S, T) = \langle V, E, S, T \rangle$.

Let x, y be sets. One can verify that $x \dot{\longmapsto} y$ is finite.

Let G be a graph structure, let n be a natural number, and let x be a set. The functor G.set(n, x) yielding a graph structure is defined as follows:

(Def. 13) G.set$(n, x) = G +\cdot (n \dot{\longmapsto} x)$.

Let G be a graph structure and let X be a set. The functor G.strict(X) yielding a graph structure is defined by:

(Def. 14) G.strict$(X) = G{\restriction}X$.

Let G be a graph. Observe that G.strict(the graph selectors) is graph-like.

Let G be a graph and let x, y, e be sets. We say that e joins x and y in G if and only if the conditions (Def. 15) are satisfied.

(Def. 15)(i) $e \in$ the edges of G, and

(ii) (the source of $G)(e) = x$ and (the target of $G)(e) = y$ or (the source of $G)(e) = y$ and (the target of $G)(e) = x$.

Let G be a graph and let x, y, e be sets. We say that e joins x to y in G if and only if:

(Def. 16) $e \in$ the edges of G and (the source of $G)(e) = x$ and (the target of $G)(e) = y$.

Let G be a graph and let X, Y, e be sets. We say that e joins a vertex from X and a vertex from Y in G if and only if the conditions (Def. 17) are satisfied.

(Def. 17)(i) $e \in$ the edges of G, and

(ii) (the source of $G)(e) \in X$ and (the target of $G)(e) \in Y$ or (the source of $G)(e) \in Y$ and (the target of $G)(e) \in X$.

We say that e joins a vertex from X to a vertex from Y in G if and only if:

(Def. 18) $e \in$ the edges of G and (the source of $G)(e) \in X$ and (the target of $G)(e) \in Y$.

Let G be a graph. We say that G is finite if and only if:

(Def. 19) The vertices of G is finite and the edges of G is finite.

We say that G is loopless if and only if:

(Def. 20) It is not true that there exists a set e such that $e \in$ the edges of G and (the source of $G)(e) =$ (the target of $G)(e)$.

We say that G is trivial if and only if:

(Def. 21) $\overline{\overline{\text{the vertices of } G}} = \mathbf{1}$.

We say that G is non-multi if and only if:

(Def. 22) For all sets e_1, e_2, v_1, v_2 such that e_1 joins v_1 and v_2 in G and e_2 joins v_1 and v_2 in G holds $e_1 = e_2$.

We say that G is non-directed-multi if and only if:

(Def. 23) For all sets e_1, e_2, v_1, v_2 such that e_1 joins v_1 to v_2 in G and e_2 joins v_1 to v_2 in G holds $e_1 = e_2$.

Let G be a graph. We say that G is simple if and only if:

(Def. 24) G is loopless and non-multi.

We say that G is directed-simple if and only if:

(Def. 25) G is loopless and non-directed-multi.

One can verify the following observations:

$*$ every graph which is non-multi is also non-directed-multi,

* every graph which is simple is also loopless and non-multi,
* every graph which is loopless and non-multi is also simple,
* every graph which is loopless and non-directed-multi is also directed-simple,
* every graph which is directed-simple is also loopless and non-directed-multi,
* every graph which is trivial and loopless is also finite, and
* every graph which is trivial and non-directed-multi is also finite.

Let us note that there exists a graph which is trivial and simple and there exists a graph which is finite, non trivial, and simple.

Let G be a finite graph. Observe that the vertices of G is finite and the edges of G is finite.

Let G be a trivial graph. One can verify that the vertices of G is finite.

Let V be a non empty finite set, let E be a finite set, and let S, T be functions from E into V. One can check that createGraph(V, E, S, T) is finite.

Let V be a non empty set, let E be an empty set, and let S, T be functions from E into V. One can check that createGraph(V, E, S, T) is simple.

Let v be a set, let E be a set, and let S, T be functions from E into $\{v\}$. Observe that createGraph$(\{v\}, E, S, T)$ is trivial.

Let G be a graph. The functor G.order() yielding a cardinal number is defined as follows:

(Def. 26) G.order() $= \overline{\overline{\text{the vertices of } G}}$.

Let G be a finite graph. Then G.order() is a non empty natural number.

Let G be a graph. The functor G.size() yields a cardinal number and is defined by:

(Def. 27) G.size() $= \overline{\overline{\text{the edges of } G}}$.

Let G be a finite graph. Then G.size() is a natural number.

Let G be a graph and let X be a set. The functor G.edgesInto(X) yields a subset of the edges of G and is defined as follows:

(Def. 28) For every set e holds $e \in G$.edgesInto(X) iff $e \in$ the edges of G and (the target of $G)(e) \in X$.

The functor G.edgesOutOf(X) yields a subset of the edges of G and is defined by:

(Def. 29) For every set e holds $e \in G$.edgesOutOf(X) iff $e \in$ the edges of G and (the source of $G)(e) \in X$.

Let G be a graph and let X be a set. The functor G.edgesInOut(X) yields a subset of the edges of G and is defined by:

(Def. 30) G.edgesInOut$(X) = G$.edgesInto$(X) \cup G$.edgesOutOf(X).

The functor G.edgesBetween(X) yielding a subset of the edges of G is defined as follows:

(Def. 31) G.edgesBetween$(X) = G$.edgesInto$(X) \cap G$.edgesOutOf(X).

Let G be a graph and let X, Y be sets. The functor G.edgesBetween(X, Y) yielding a subset of the edges of G is defined by:

(Def. 32) For every set e holds $e \in G$.edgesBetween(X, Y) iff e joins a vertex from X and a vertex from Y in G.

The functor G.edgesDBetween(X, Y) yields a subset of the edges of G and is defined as follows:

(Def. 33) For every set e holds $e \in G$.edgesDBetween(X, Y) iff e joins a vertex from X to a vertex from Y in G.

In this article we present several logical schemes. The scheme *FinGraphOrderInd* concerns a unary predicate $\mathcal{P}$, and states that:

For every finite graph G holds $\mathcal{P}[G]$

provided the following conditions are met:

- For every finite graph G such that G.order() $= 1$ holds $\mathcal{P}[G]$, and
- Let k be a non empty natural number. Suppose that for every finite graph G_1 such that G_1.order() $= k$ holds $\mathcal{P}[G_1]$. Let G_2 be a finite graph. If G_2.order() $= k + 1$, then $\mathcal{P}[G_2]$.

The scheme *FinGraphSizeInd* concerns a unary predicate $\mathcal{P}$, and states that:

For every finite graph G holds $\mathcal{P}[G]$

provided the following requirements are met:

- For every finite graph G such that G.size() $= 0$ holds $\mathcal{P}[G]$, and
- Let k be a natural number. Suppose that for every finite graph G_1 such that G_1.size() $= k$ holds $\mathcal{P}[G_1]$. Let G_2 be a finite graph. If G_2.size() $= k + 1$, then $\mathcal{P}[G_2]$.

Let G be a graph. A graph is called a subgraph of G if it satisfies the conditions (Def. 34).

(Def. 34)(i) The vertices of it $\subseteq$ the vertices of G,

(ii) the edges of it $\subseteq$ the edges of G, and

(iii) for every set e such that $e \in$ the edges of it holds (the source of it)$(e) =$ (the source of G)(e) and (the target of it)$(e) =$ (the target of G)(e).

Let G_3 be a graph and let G_4 be a subgraph of G_3. Then the vertices of G_4 is a non empty subset of the vertices of G_3. Then the edges of G_4 is a subset of the edges of G_3.

Let G be a graph. Note that there exists a subgraph of G which is trivial and simple.

Let G be a finite graph. Note that every subgraph of G is finite.

Let G be a loopless graph. Observe that every subgraph of G is loopless.

Let G be a trivial graph. One can check that every subgraph of G is trivial.

Let G be a non-multi graph. Observe that every subgraph of G is non-multi.

Let G_3 be a graph and let G_4 be a subgraph of G_3. We say that G_4 is spanning if and only if:

(Def. 35) The vertices of $G_4 =$ the vertices of G_3.

Let G be a graph. One can verify that there exists a subgraph of G which is spanning.

Let G_3, G_4 be graphs. The predicate $G_3 =_G G_4$ is defined by the conditions (Def. 36).

(Def. 36)(i) The vertices of $G_3 =$ the vertices of G_4,

(ii) the edges of $G_3 =$ the edges of G_4,

(iii) the source of $G_3 =$ the source of G_4, and

(iv) the target of $G_3 =$ the target of G_4.

Let us notice that the predicate $G_3 =_G G_4$ is reflexive and symmetric.

Let G_3, G_4 be graphs. We introduce $G_3 \neq_G G_4$ as an antonym of $G_3 =_G G_4$.

Let G_3, G_4 be graphs. The predicate $G_3 \subseteq G_4$ is defined as follows:

(Def. 37) G_3 is a subgraph of G_4.

Let us note that the predicate $G_3 \subseteq G_4$ is reflexive.

Let G_3, G_4 be graphs. The predicate $G_3 \subset G_4$ is defined as follows:

(Def. 38) $G_3 \subseteq G_4$ and $G_3 \neq_G G_4$.

Let us note that the predicate $G_3 \subset G_4$ is irreflexive.

Let G be a graph and let V, E be sets. A subgraph of G is called a subgraph of G induced by V and E if:

(Def. 39)(i) The vertices of it $= V$ and the edges of it $= E$ if V is a non empty subset of the vertices of G and $E \subseteq G$.edgesBetween(V),

(ii) it $=_G G$, otherwise.

Let G be a graph and let V be a set. A subgraph of G induced by V is a subgraph of G induced by V and G.edgesBetween(V).

Let G be a graph, let V be a finite non empty subset of the vertices of G, and let E be a finite subset of G.edgesBetween(V). Observe that every subgraph of G induced by V and E is finite.

Let G be a graph, let v be an element of the vertices of G, and let E be a subset of G.edgesBetween($\{v\}$). Note that every subgraph of G induced by $\{v\}$ and E is trivial.

Let G be a graph and let v be an element of the vertices of G. Note that every subgraph of G induced by $\{v\}$ and $\emptyset$ is finite and trivial.

Let G be a graph and let V be a non empty subset of the vertices of G. Note that every subgraph of G induced by V and $\emptyset$ is simple.

Let G be a graph and let E be a subset of the edges of G. Observe that every subgraph of G induced by the vertices of G and E is spanning.

Let G be a graph. One can check that every subgraph of G induced by the vertices of G and $\emptyset$ is spanning.

Let G be a graph and let v be a set. A subgraph of G with vertex v removed is a subgraph of G induced by (the vertices of G) $\setminus \{v\}$.

Let G be a graph and let V be a set. A subgraph of G with vertices V removed is a subgraph of G induced by (the vertices of G) $\setminus V$.

Let G be a graph and let e be a set. A subgraph of G with edge e removed is a subgraph of G induced by the vertices of G and (the edges of G) $\setminus \{e\}$.

Let G be a graph and let E be a set. A subgraph of G with edges E removed is a subgraph of G induced by the vertices of G and (the edges of G) $\setminus E$.

Let G be a graph and let e be a set. Observe that every subgraph of G with edge e removed is spanning.

Let G be a graph and let E be a set. Observe that every subgraph of G with edges E removed is spanning.

Let G be a graph. A vertex of G is an element of the vertices of G.

Let G be a graph and let v be a vertex of G. The functor v.edgesIn() yielding a subset of the edges of G is defined as follows:

(Def. 40) $v.\mathrm{edgesIn}() = G.\mathrm{edgesInto}(\{v\})$.

The functor v.edgesOut() yields a subset of the edges of G and is defined as follows:

(Def. 41) $v.\mathrm{edgesOut}() = G.\mathrm{edgesOutOf}(\{v\})$.

The functor v.edgesInOut() yields a subset of the edges of G and is defined by:

(Def. 42) $v.\mathrm{edgesInOut}() = G.\mathrm{edgesInOut}(\{v\})$.

Let G be a graph, let v be a vertex of G, and let e be a set. The functor v.adj(e) yields a vertex of G and is defined by:

(Def. 43) $$v.\mathrm{adj}(e) = \begin{cases} (\text{the source of } G)(e), & \text{if } e \in \text{the edges of } G \text{ and} \\ \quad (\text{the target of } G)(e) = v, \\ (\text{the target of } G)(e), & \text{if } e \in \text{the edges of } G \text{ and} \\ \quad (\text{the source of } G)(e) = v \text{ and } (\text{the target of } G)(e) \neq v, \\ v, & \text{otherwise.} \end{cases}$$

Let G be a graph and let v be a vertex of G. The functor v.inDegree() yields a cardinal number and is defined as follows:

(Def. 44) $v.\mathrm{inDegree}() = \overline{\overline{v.\mathrm{edgesIn}()}}$.

The functor v.outDegree() yielding a cardinal number is defined as follows:

(Def. 45) $v.\mathrm{outDegree}() = \overline{\overline{v.\mathrm{edgesOut}()}}$.

Let G be a finite graph and let v be a vertex of G. Then v.inDegree() is a natural number. Then v.outDegree() is a natural number.

Let G be a graph and let v be a vertex of G. The functor v.degree() yielding a cardinal number is defined as follows:

(Def. 46) $v.\mathrm{degree}() = v.\mathrm{inDegree}() + v.\mathrm{outDegree}()$.

Let G be a finite graph and let v be a vertex of G. Then v.degree() is a natural number and it can be characterized by the condition:

(Def. 47) v.degree() = v.inDegree() + v.outDegree().

Let G be a graph and let v be a vertex of G. The functor v.inNeighbors() yields a subset of the vertices of G and is defined as follows:

(Def. 48) v.inNeighbors() = (the source of G)$^\circ v$.edgesIn().

The functor v.outNeighbors() yielding a subset of the vertices of G is defined by:

(Def. 49) v.outNeighbors() = (the target of G)$^\circ v$.edgesOut().

Let G be a graph and let v be a vertex of G. The functor v.allNeighbors() yields a subset of the vertices of G and is defined by:

(Def. 50) v.allNeighbors() = v.inNeighbors() $\cup$ v.outNeighbors().

Let G be a graph and let v be a vertex of G. We say that v is isolated if and only if:

(Def. 51) v.edgesInOut() = $\emptyset$.

Let G be a finite graph and let v be a vertex of G. Let us observe that v is isolated if and only if:

(Def. 52) v.degree() = 0.

Let G be a graph and let v be a vertex of G. We say that v is endvertex if and only if:

(Def. 53) There exists a set e such that v.edgesInOut() = $\{e\}$ and e does not join v and v in G.

Let G be a finite graph and let v be a vertex of G. Let us observe that v is endvertex if and only if:

(Def. 54) v.degree() = 1.

Let F be a many sorted set indexed by $\mathbb{N}$. We say that F is graph-yielding if and only if:

(Def. 55) For every natural number n holds $F(n)$ is a graph.

We say that F is halting if and only if:

(Def. 56) There exists a natural number n such that $F(n) = F(n+1)$.

Let F be a many sorted set indexed by $\mathbb{N}$. The functor F.Lifespan() yielding a natural number is defined by:

(Def. 57)(i) $F(F$.Lifespan()$) = F(F$.Lifespan()$+1)$ and for every natural number n such that $F(n) = F(n+1)$ holds F.Lifespan() $\leq n$ if F is halting,

(ii) F.Lifespan() = 0, otherwise.

Let F be a many sorted set indexed by $\mathbb{N}$. The functor F.Result() yielding a set is defined by:

(Def. 58) F.Result() = $F(F$.Lifespan()$)$.

Let us mention that there exists a many sorted set indexed by $\mathbb{N}$ which is graph-yielding.

A graph sequence is a graph-yielding many sorted set indexed by $\mathbb{N}$.

Let G_5 be a graph sequence and let x be a natural number. The functor $G_5.{\rightarrow}x$ yields a graph and is defined by:

(Def. 59) $G_5.{\rightarrow}x = G_5(x)$.

Let G_5 be a graph sequence. We say that G_5 is finite if and only if:

(Def. 60) For every natural number x holds $G_5.{\rightarrow}x$ is finite.

We say that G_5 is loopless if and only if:

(Def. 61) For every natural number x holds $G_5.{\rightarrow}x$ is loopless.

We say that G_5 is trivial if and only if:

(Def. 62) For every natural number x holds $G_5.{\rightarrow}x$ is trivial.

We say that G_5 is non-trivial if and only if:

(Def. 63) For every natural number x holds $G_5.{\rightarrow}x$ is non trivial.

We say that G_5 is non-multi if and only if:

(Def. 64) For every natural number x holds $G_5.{\rightarrow}x$ is non-multi.

We say that G_5 is non-directed-multi if and only if:

(Def. 65) For every natural number x holds $G_5.{\rightarrow}x$ is non-directed-multi.

We say that G_5 is simple if and only if:

(Def. 66) For every natural number x holds $G_5.{\rightarrow}x$ is simple.

We say that G_5 is directed-simple if and only if:

(Def. 67) For every natural number x holds $G_5.{\rightarrow}x$ is directed-simple.

Let G_5 be a graph sequence. Let us observe that G_5 is halting if and only if:

(Def. 68) There exists a natural number n such that $G_5.{\rightarrow}n = G_5.{\rightarrow}(n + 1)$.

One can verify that there exists a graph sequence which is halting, finite, loopless, trivial, non-multi, non-directed-multi, simple, and directed-simple and there exists a graph sequence which is halting, finite, loopless, non-trivial, non-multi, non-directed-multi, simple, and directed-simple.

Let G_5 be a finite graph sequence and let x be a natural number. One can check that $G_5.{\rightarrow}x$ is finite.

Let G_5 be a loopless graph sequence and let x be a natural number. Note that $G_5.{\rightarrow}x$ is loopless.

Let G_5 be a trivial graph sequence and let x be a natural number. Observe that $G_5.{\rightarrow}x$ is trivial.

Let G_5 be a non-trivial graph sequence and let x be a natural number. Observe that $G_5.{\rightarrow}x$ is non trivial.

Let G_5 be a non-multi graph sequence and let x be a natural number. Note that $G_5.{\rightarrow}x$ is non-multi.

Let G_5 be a non-directed-multi graph sequence and let x be a natural number. Observe that $G_5.{\rightarrow}x$ is non-directed-multi.

Let G_5 be a simple graph sequence and let x be a natural number. Note that $G_5 \hookrightarrow x$ is simple.

Let G_5 be a directed-simple graph sequence and let x be a natural number. Note that $G_5 \hookrightarrow x$ is directed-simple.

One can check that every graph sequence which is non-multi is also non-directed-multi.

Let us observe that every graph sequence which is simple is also loopless and non-multi.

One can verify that every graph sequence which is loopless and non-multi is also simple.

Let us note that every graph sequence which is loopless and non-directed-multi is also directed-simple.

One can verify that every graph sequence which is directed-simple is also loopless and non-directed-multi.

Let us note that every graph sequence which is trivial and loopless is also finite.

Let us observe that every graph sequence which is trivial and non-directed-multi is also finite.

2. Theorems

For simplicity, we adopt the following convention: G_6 denotes a graph structure, G, G_3, G_4, G_7 denote graphs, e, x, x_1, x_2, y, y_1, y_2, E, V, X, Y denote sets, n, n_1, n_2 denote natural numbers, and v, v_1, v_2 denote vertices of G.

We now state a number of propositions:

(1) VertexSelector $= 1$ and EdgeSelector $= 2$ and SourceSelector $= 3$ and TargetSelector $= 4$.

(2) $x \in$ the graph selectors iff $x =$ VertexSelector or $x =$ EdgeSelector or $x =$ SourceSelector or $x =$ TargetSelector .

(3) The graph selectors $\subseteq \operatorname{dom} G$.

(4) The vertices of $G_6 = G_6$(VertexSelector) and the edges of $G_6 = G_6$(EdgeSelector) and the source of $G_6 = G_6$(SourceSelector) and the target of $G_6 = G_6$(TargetSelector).

(5)(i) dom (the source of G) = the edges of G,

(ii) dom (the target of G) = the edges of G,

(iii) rng (the source of G) $\subseteq$ the vertices of G, and

(iv) rng (the target of G) $\subseteq$ the vertices of G.

(7)[3] G_6 is graph-like if and only if the following conditions are satisfied:

(i) the graph selectors $\subseteq \operatorname{dom} G_6$,

[3]The proposition (6) has been removed.

(ii) the vertices of G_6 is non empty,

(iii) the source of G_6 is a function from the edges of G_6 into the vertices of G_6, and

(iv) the target of G_6 is a function from the edges of G_6 into the vertices of G_6.

(8) Let V be a non empty set, E be a set, and S, T be functions from E into V. Then

(i) the vertices of createGraph$(V, E, S, T) = V$,

(ii) the edges of createGraph$(V, E, S, T) = E$,

(iii) the source of createGraph$(V, E, S, T) = S$, and

(iv) the target of createGraph$(V, E, S, T) = T$.

(9) $\mathrm{dom}(G_6.\mathrm{set}(n, x)) = \mathrm{dom}\, G_6 \cup \{n\}$.

(10) $\mathrm{dom}\, G_6 \subseteq \mathrm{dom}(G_6.\mathrm{set}(n, x))$.

(11) $(G_6.\mathrm{set}(n, x))(n) = x$.

(12) If $n_1 \neq n_2$, then $G_6(n_2) = (G_6.\mathrm{set}(n_1, x))(n_2)$.

(13) Suppose $n \notin$ the graph selectors. Then

(i) the vertices of $G =$ the vertices of $G.\mathrm{set}(n, x)$,

(ii) the edges of $G =$ the edges of $G.\mathrm{set}(n, x)$,

(iii) the source of $G =$ the source of $G.\mathrm{set}(n, x)$,

(iv) the target of $G =$ the target of $G.\mathrm{set}(n, x)$, and

(v) $G.\mathrm{set}(n, x)$ is a graph.

(14) The vertices of $G_6.\mathrm{set}(\mathrm{VertexSelector}, x) = x$ and the edges of $G_6.\mathrm{set}(\mathrm{EdgeSelector}, x) = x$ and the source of $G_6.\mathrm{set}(\mathrm{SourceSelector}, x) = x$ and the target of $G_6.\mathrm{set}(\mathrm{TargetSelector}, x) = x$.

(15) If $n_1 \neq n_2$, then $n_1 \in \mathrm{dom}(G_6.\mathrm{set}(n_1, x).\mathrm{set}(n_2, y))$ and $n_2 \in \mathrm{dom}(G_6.\mathrm{set}(n_1, x).\mathrm{set}(n_2, y))$ and $(G_6.\mathrm{set}(n_1, x).\mathrm{set}(n_2, y))(n_1) = x$ and $(G_6.\mathrm{set}(n_1, x).\mathrm{set}(n_2, y))(n_2) = y$.

(16) If e joins x and y in G, then $x \in$ the vertices of G and $y \in$ the vertices of G.

(17) If e joins x and y in G, then e joins y and x in G.

(18) If e joins x_1 and y_1 in G and e joins x_2 and y_2 in G, then $x_1 = x_2$ and $y_1 = y_2$ or $x_1 = y_2$ and $y_1 = x_2$.

(19) e joins x and y in G iff e joins x to y in G or e joins y to x in G.

(20) Suppose e joins x and y in G but $x \in X$ and $y \in Y$ or $x \in Y$ and $y \in X$. Then e joins a vertex from X and a vertex from Y in G.

(21) G is loopless iff for every set v it is not true that there exists a set e such that e joins v and v in G.

(22) For every finite loopless graph G and for every vertex v of G holds $v.\mathrm{degree}() = \mathrm{card}(v.\mathrm{edgesInOut}())$.

(23) For every non trivial graph G and for every vertex v of G holds (the vertices of G) $\setminus \{v\}$ is non empty.

(24) For every non trivial graph G there exist vertices v_1, v_2 of G such that $v_1 \neq v_2$.

(25) For every trivial graph G there exists a vertex v of G such that the vertices of $G = \{v\}$.

(26) For every trivial loopless graph G holds the edges of $G = \emptyset$.

(27) If the edges of $G = \emptyset$, then G is simple.

(28) For every finite graph G holds G.order() ≥ 1.

(29) For every finite graph G holds G.order() $= 1$ iff G is trivial.

(30) For every finite graph G holds G.order() $= 1$ iff there exists a vertex v of G such that the vertices of $G = \{v\}$.

(31) $e \in$ the edges of G but (the source of G)$(e) \in X$ or (the target of G)$(e) \in X$ iff $e \in G$.edgesInOut(X).

(32) G.edgesInto$(X) \subseteq G$.edgesInOut(X) and G.edgesOutOf$(X) \subseteq G$.edgesInOut(X).

(33) The edges of $G = G$.edgesInOut(the vertices of G).

(34) $e \in$ the edges of G and (the source of G)$(e) \in X$ and (the target of G)$(e) \in X$ iff $e \in G$.edgesBetween(X).

(35) If $x \in X$ and $y \in X$ and e joins x and y in G, then $e \in G$.edgesBetween(X).

(36) G.edgesBetween$(X) \subseteq G$.edgesInOut(X).

(37) The edges of $G = G$.edgesBetween(the vertices of G).

(38) (The edges of G) $\setminus G$.edgesInOut$(X) = G$.edgesBetween((the vertices of G) $\setminus X$).

(39) If $X \subseteq Y$, then G.edgesBetween$(X) \subseteq G$.edgesBetween(Y).

(40) For every graph G and for all sets X_1, X_2, Y_1, Y_2 such that $X_1 \subseteq X_2$ and $Y_1 \subseteq Y_2$ holds G.edgesBetween$(X_1, Y_1) \subseteq G$.edgesBetween(X_2, Y_2).

(41) For every graph G and for all sets X_1, X_2, Y_1, Y_2 such that $X_1 \subseteq X_2$ and $Y_1 \subseteq Y_2$ holds G.edgesDBetween$(X_1, Y_1) \subseteq G$.edgesDBetween(X_2, Y_2).

(42) For every graph G and for every vertex v of G holds v.edgesIn() $= G$.edgesDBetween(the vertices of G, $\{v\}$) and v.edgesOut() $= G$.edgesDBetween$(\{v\}$, the vertices of G).

(43) G is a subgraph of G.

(44) G_3 is a subgraph of G_4 and G_4 is a subgraph of G_3 if and only if the following conditions are satisfied:

(i) the vertices of $G_3 =$ the vertices of G_4,

(ii) the edges of $G_3 =$ the edges of G_4,

(iii) the source of $G_3 =$ the source of G_4, and

(iv) the target of G_3 = the target of G_4.

(45) Let G_3 be a graph, G_4 be a subgraph of G_3, and x be a set. Then

(i) if $x \in$ the vertices of G_4, then $x \in$ the vertices of G_3, and

(ii) if $x \in$ the edges of G_4, then $x \in$ the edges of G_3.

(46) For every graph G_3 and for every subgraph G_4 of G_3 holds every subgraph of G_4 is a subgraph of G_3.

(47) Let G be a graph and G_3, G_4 be subgraphs of G. Suppose the vertices of $G_3 \subseteq$ the vertices of G_4 and the edges of $G_3 \subseteq$ the edges of G_4. Then G_3 is a subgraph of G_4.

(48) Let G_3 be a graph and G_4 be a subgraph of G_3. Then

(i) the source of G_4 = (the source of G_3)$\restriction$(the edges of G_4), and

(ii) the target of G_4 = (the target of G_3)$\restriction$(the edges of G_4).

(49) Let G be a graph, V_1, V_2, E_1, E_2 be sets, G_3 be a subgraph of G induced by V_1 and E_1, and G_4 be a subgraph of G induced by V_2 and E_2. Suppose $V_2 \subseteq V_1$ and $E_2 \subseteq E_1$ and V_2 is a non empty subset of the vertices of G and $E_2 \subseteq G$.edgesBetween(V_2). Then G_4 is a subgraph of G_3.

(50) Let G_3 be a non trivial graph, v be a vertex of G_3, and G_4 be a subgraph of G_3 with vertex v removed. Then the vertices of G_4 = (the vertices of G_3) $\setminus \{v\}$ and the edges of G_4 = G_3.edgesBetween((the vertices of G_3) $\setminus \{v\}$).

(51) Let G_3 be a finite non trivial graph, v be a vertex of G_3, and G_4 be a subgraph of G_3 with vertex v removed. Then G_4.order() + 1 = G_3.order() and G_4.size() + card(v.edgesInOut()) = G_3.size().

(52) Let G_3 be a graph, V be a set, and G_4 be a subgraph of G_3 with vertices V removed. Suppose $V \subset$ the vertices of G_3. Then the vertices of G_4 = (the vertices of G_3) $\setminus V$ and the edges of G_4 = G_3.edgesBetween((the vertices of G_3) $\setminus V$).

(53) Let G_3 be a finite graph, V be a subset of the vertices of G_3, and G_4 be a subgraph of G_3 with vertices V removed. If $V \neq$ the vertices of G_3, then G_4.order() + card V = G_3.order() and G_4.size() + card(G_3.edgesInOut(V)) = G_3.size().

(54) Let G_3 be a graph, e be a set, and G_4 be a subgraph of G_3 with edge e removed. Then the vertices of G_4 = the vertices of G_3 and the edges of G_4 = (the edges of G_3) $\setminus \{e\}$.

(55) Let G_3 be a finite graph, e be a set, and G_4 be a subgraph of G_3 with edge e removed. Then G_3.order() = G_4.order() and if $e \in$ the edges of G_3, then G_4.size() + 1 = G_3.size().

(56) Let G_3 be a graph, E be a set, and G_4 be a subgraph of G_3 with edges E removed. Then the vertices of G_4 = the vertices of G_3 and the edges of G_4 = (the edges of G_3) $\setminus E$.

(57) For every finite graph G_3 and for every set E and for every subgraph G_4 of G_3 with edges E removed holds G_3.order() $= G_4$.order().

(58) Let G_3 be a finite graph, E be a subset of the edges of G_3, and G_4 be a subgraph of G_3 with edges E removed. Then G_4.size() $+ \operatorname{card} E =$ G_3.size().

(59) $e \in v$.edgesIn() iff $e \in$ the edges of G and (the target of $G)(e) = v$.

(60) $e \in v$.edgesIn() iff there exists a set x such that e joins x to v in G.

(61) $e \in v$.edgesOut() iff $e \in$ the edges of G and (the source of $G)(e) = v$.

(62) $e \in v$.edgesOut() iff there exists a set x such that e joins v to x in G.

(63) v.edgesInOut() $= v$.edgesIn() $\cup\ v$.edgesOut().

(64) $e \in v$.edgesInOut() iff $e \in$ the edges of G but (the source of $G)(e) = v$ or (the target of $G)(e) = v$.

(65) If e joins v_1 and x in G, then $e \in v_1$.edgesInOut().

(66) If e joins v_1 and v_2 in G, then $e \in v_1$.edgesIn() and $e \in v_2$.edgesOut() or $e \in v_2$.edgesIn() and $e \in v_1$.edgesOut().

(67) $e \in v_1$.edgesInOut() iff there exists a vertex v_2 of G such that e joins v_1 and v_2 in G.

(68) If $e \in v$.edgesInOut() and e joins x and y in G, then $v = x$ or $v = y$.

(69) If e joins v_1 and v_2 in G, then v_1.adj$(e) = v_2$ and v_2.adj$(e) = v_1$.

(70) $e \in v$.edgesInOut() iff e joins v and v.adj(e) in G.

(71) Let G be a finite graph, e be a set, and v_1, v_2 be vertices of G. If e joins v_1 and v_2 in G, then $1 \leq v_1$.degree() and $1 \leq v_2$.degree().

(72) $x \in v$.inNeighbors() iff there exists a set e such that e joins x to v in G.

(73) $x \in v$.outNeighbors() iff there exists a set e such that e joins v to x in G.

(74) $x \in v$.allNeighbors() iff there exists a set e such that e joins v and x in G.

(75) Let G_3 be a graph, G_4 be a subgraph of G_3, and x, y, e be sets. Then
 (i) if e joins x and y in G_4, then e joins x and y in G_3,
 (ii) if e joins x to y in G_4, then e joins x to y in G_3,
 (iii) if e joins a vertex from x and a vertex from y in G_4, then e joins a vertex from x and a vertex from y in G_3, and
 (iv) if e joins a vertex from x to a vertex from y in G_4, then e joins a vertex from x to a vertex from y in G_3.

(76) Let G_3 be a graph, G_4 be a subgraph of G_3, and x, y, e be sets such that $e \in$ the edges of G_4. Then
 (i) if e joins x and y in G_3, then e joins x and y in G_4,
 (ii) if e joins x to y in G_3, then e joins x to y in G_4,

(iii) if e joins a vertex from x and a vertex from y in G_3, then e joins a vertex from x and a vertex from y in G_4, and

(iv) if e joins a vertex from x to a vertex from y in G_3, then e joins a vertex from x to a vertex from y in G_4.

(77) For every graph G_3 and for every spanning subgraph G_4 of G_3 holds every spanning subgraph of G_4 is a spanning subgraph of G_3.

(78) For every finite graph G_3 and for every subgraph G_4 of G_3 holds G_4.order() $\leq G_3$.order() and G_4.size() $\leq G_3$.size().

(79) Let G_3 be a graph, G_4 be a subgraph of G_3, and X be a set. Then G_4.edgesInto(X) $\subseteq G_3$.edgesInto(X) and G_4.edgesOutOf(X) $\subseteq G_3$.edgesOutOf(X) and G_4.edgesInOut(X) $\subseteq G_3$.edgesInOut(X) and G_4.edgesBetween(X) $\subseteq G_3$.edgesBetween(X).

(80) For every graph G_3 and for every subgraph G_4 of G_3 and for all sets X, Y holds G_4.edgesBetween(X, Y) $\subseteq G_3$.edgesBetween(X, Y) and G_4.edgesDBetween(X, Y) $\subseteq G_3$.edgesDBetween(X, Y).

(81) Let G_3 be a graph, G_4 be a subgraph of G_3, v_1 be a vertex of G_3, and v_2 be a vertex of G_4. If $v_1 = v_2$, then v_2.edgesIn() $\subseteq v_1$.edgesIn() and v_2.edgesOut() $\subseteq v_1$.edgesOut() and v_2.edgesInOut() $\subseteq v_1$.edgesInOut().

(82) Let G_3 be a graph, G_4 be a subgraph of G_3, v_1 be a vertex of G_3, and v_2 be a vertex of G_4. Suppose $v_1 = v_2$. Then v_2.edgesIn() $= v_1$.edgesIn() $\cap$ the edges of G_4 and v_2.edgesOut() $= v_1$.edgesOut() $\cap$ the edges of G_4 and v_2.edgesInOut() $= v_1$.edgesInOut() $\cap$ the edges of G_4.

(83) Let G_3 be a graph, G_4 be a subgraph of G_3, v_1 be a vertex of G_3, v_2 be a vertex of G_4, and e be a set. If $v_1 = v_2$ and $e \in$ the edges of G_4, then v_1.adj(e) $= v_2$.adj(e).

(84) Let G_3 be a finite graph, G_4 be a subgraph of G_3, v_1 be a vertex of G_3, and v_2 be a vertex of G_4. If $v_1 = v_2$, then v_2.inDegree() $\leq v_1$.inDegree() and v_2.outDegree() $\leq v_1$.outDegree() and v_2.degree() $\leq v_1$.degree().

(85) Let G_3 be a graph, G_4 be a subgraph of G_3, v_1 be a vertex of G_3, and v_2 be a vertex of G_4. If $v_1 = v_2$, then v_2.inNeighbors() $\subseteq v_1$.inNeighbors() and v_2.outNeighbors() $\subseteq v_1$.outNeighbors() and v_2.allNeighbors() $\subseteq v_1$.allNeighbors().

(86) Let G_3 be a graph, G_4 be a subgraph of G_3, v_1 be a vertex of G_3, and v_2 be a vertex of G_4. If $v_1 = v_2$ and v_1 is isolated, then v_2 is isolated.

(87) Let G_3 be a graph, G_4 be a subgraph of G_3, v_1 be a vertex of G_3, and v_2 be a vertex of G_4. If $v_1 = v_2$ and v_1 is endvertex, then v_2 is endvertex or isolated.

(88) If $G_3 =_G G_4$ and $G_4 =_G G_7$, then $G_3 =_G G_7$.

(89) Let G be a graph and G_3, G_4 be subgraphs of G. Suppose the vertices of $G_3 =$ the vertices of G_4 and the edges of $G_3 =$ the edges of G_4. Then

$G_3 =_G G_4$.

(90) $G_3 =_G G_4$ iff G_3 is a subgraph of G_4 and G_4 is a subgraph of G_3.

(91) Suppose $G_3 =_G G_4$. Then

(i) if e joins x and y in G_3, then e joins x and y in G_4,

(ii) if e joins x to y in G_3, then e joins x to y in G_4,

(iii) if e joins a vertex from X and a vertex from Y in G_3, then e joins a vertex from X and a vertex from Y in G_4, and

(iv) if e joins a vertex from X to a vertex from Y in G_3, then e joins a vertex from X to a vertex from Y in G_4.

(92) Suppose $G_3 =_G G_4$. Then

(i) if G_3 is finite, then G_4 is finite,

(ii) if G_3 is loopless, then G_4 is loopless,

(iii) if G_3 is trivial, then G_4 is trivial,

(iv) if G_3 is non-multi, then G_4 is non-multi,

(v) if G_3 is non-directed-multi, then G_4 is non-directed-multi,

(vi) if G_3 is simple, then G_4 is simple, and

(vii) if G_3 is directed-simple, then G_4 is directed-simple.

(93) If $G_3 =_G G_4$, then G_3.order() = G_4.order() and G_3.size() = G_4.size() and G_3.edgesInto(X) = G_4.edgesInto(X) and G_3.edgesOutOf(X) = G_4.edgesOutOf(X) and G_3.edgesInOut(X) = G_4.edgesInOut(X) and G_3.edgesBetween(X) = G_4.edgesBetween(X) and G_3.edgesDBetween(X, Y) = G_4.edgesDBetween(X, Y).

(94) If $G_3 =_G G_4$ and G_7 is a subgraph of G_3, then G_7 is a subgraph of G_4.

(95) If $G_3 =_G G_4$ and G_3 is a subgraph of G_7, then G_4 is a subgraph of G_7.

(96) For all subgraphs G_3, G_4 of G induced by V and E holds $G_3 =_G G_4$.

(97) For every graph G_3 and for every subgraph G_4 of G_3 induced by the vertices of G_3 holds $G_3 =_G G_4$.

(98) Let G_3, G_4 be graphs, V, E be sets, and G_7 be a subgraph of G_3 induced by V and E. If $G_3 =_G G_4$, then G_7 is a subgraph of G_4 induced by V and E.

(99) Let v_1 be a vertex of G_3 and v_2 be a vertex of G_4. Suppose $v_1 = v_2$ and $G_3 =_G G_4$. Then v_1.edgesIn() = v_2.edgesIn() and v_1.edgesOut() = v_2.edgesOut() and v_1.edgesInOut() = v_2.edgesInOut() and v_1.adj(e) = v_2.adj(e) and v_1.inDegree() = v_2.inDegree() and v_1.outDegree() = v_2.outDegree() and v_1.degree() = v_2.degree() and v_1.inNeighbors() = v_2.inNeighbors() and v_1.outNeighbors() = v_2.outNeighbors() and v_1.allNeighbors() = v_2.allNeighbors().

(100) Let v_1 be a vertex of G_3 and v_2 be a vertex of G_4 such that $v_1 = v_2$ and $G_3 =_G G_4$. Then

(i) if v_1 is isolated, then v_2 is isolated, and

(ii) if v_1 is endvertex, then v_2 is endvertex.

(101) Let G be a graph and G_3, G_4 be subgraphs of G. Suppose $G_3 \subset G_4$. Then the vertices of $G_3 \subset$ the vertices of G_4 or the edges of $G_3 \subset$ the edges of G_4.

(102) Let G be a graph and G_3, G_4 be subgraphs of G. Suppose $G_3 \subset G_4$. Then

(i) there exists a set v such that $v \in$ the vertices of G_4 and $v \notin$ the vertices of G_3, or

(ii) there exists a set e such that $e \in$ the edges of G_4 and $e \notin$ the edges of G_3.

References

[1] Grzegorz Bancerek. Cardinal arithmetics. *Formalized Mathematics*, 1(**3**):543–547, 1990.
[2] Grzegorz Bancerek. Cardinal numbers. *Formalized Mathematics*, 1(**2**):377–382, 1990.
[3] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.
[4] Grzegorz Bancerek. Sequences of ordinal numbers. *Formalized Mathematics*, 1(**2**):281–290, 1990.
[5] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.
[6] Czesław Byliński. A classical first order language. *Formalized Mathematics*, 1(**4**):669–676, 1990.
[7] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.
[8] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.
[9] Czesław Byliński. The modification of a function by a function and the iteration of the composition of a function. *Formalized Mathematics*, 1(**3**):521–527, 1990.
[10] Czesław Byliński and Piotr Rudnicki. Bounding boxes for compact sets in $\mathcal{E}^2$. *Formalized Mathematics*, 6(**3**):427–440, 1997.
[11] Jing-Chao Chen. A small computer model with push-down stack. *Formalized Mathematics*, 8(**1**):175–182, 1999.
[12] Agata Darmochwał. Finite sets. *Formalized Mathematics*, 1(**1**):165–167, 1990.
[13] Krzysztof Hryniewiecki. Graphs. *Formalized Mathematics*, 2(**3**):365–370, 1991.
[14] Gilbert Lee. *Verification of graph algorithms in Mizar*. Dept. of Comp. Sci., University of Alberta, Edmonton, Canada, 2004. M Sc thesis, `http://www.cs.ualberta.ca/-~piotr/Mizar/Doc/GL-thesis.ps`.
[15] Gilbert Lee. Trees and Graph Components. *Formalized Mathematics*, 13(**2**):271–277, 2005.
[16] Gilbert Lee. Walks in Graphs. *Formalized Mathematics*, 13(**2**):253–269, 2005.
[17] Gilbert Lee. Weighted and Labeled Graphs. *Formalized Mathematics*, 13(**2**):279–293, 2005.
[18] Andrzej Trybulec. Subsets of complex numbers. *To appear in Formalized Mathematics.*
[19] Andrzej Trybulec. Domains and their Cartesian products. *Formalized Mathematics*, 1(**1**):115–122, 1990.
[20] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[21] Andrzej Trybulec. Many-sorted sets. *Formalized Mathematics*, 4(**1**):15–22, 1993.
[22] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[23] Josef Urban. Basic facts about inaccessible and measurable cardinals. *Formalized Mathematics*, 9(**2**):323–329, 2001.

[24] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.
[25] Edmund Woronowicz. Relations defined on sets. *Formalized Mathematics*, 1(**1**):181–186, 1990.

Received February 22, 2005