FORMALIZED MATHEMATICS
Volume 13, Number 2, Pages 295–304
University of Białystok, 2005

Correctness of Dijkstra's Shortest Path and Prim's Minimum Spanning Tree Algorithms[1]

Gilbert Lee[2]
University of Victoria
Victoria, Canada

Piotr Rudnicki
University of Alberta
Edmonton, Canada

Summary. We prove correctness for Dijkstra's shortest path algorithm and Prim's minimum weight spanning tree algorithm at the level of graph manipulations.

MML identifier: `GLIB_004`, version: `7.5.01 4.39.921`

The notation and terminology used in this paper are introduced in the following articles: [25], [11], [24], [22], [28], [23], [13], [30], [10], [7], [4], [6], [14], [1], [26], [29], [8], [3], [27], [21], [19], [12], [2], [5], [9], [18], [16], [15], [20], and [17].

1. Preliminaries

One can prove the following propositions:

(1) For all functions f, g holds $\mathrm{support}(f +\cdot g) \subseteq \mathrm{support}\, f \cup \mathrm{support}\, g$.

(2) For every function f and for all sets x, y holds $\mathrm{support}(f +\cdot (x \dot{\longmapsto} y)) \subseteq \mathrm{support}\, f \cup \{x\}$.

(3) Let A, B be sets, b be a real bag over A, b_1 be a real bag over B, and b_2 be a real bag over $A \setminus B$. If $b = b_1 +\cdot b_2$, then $\sum b = \sum b_1 + \sum b_2$.

(4) For all sets X, x and for every real bag b over X such that $\mathrm{dom}\, b = \{x\}$ holds $\sum b = b(x)$.

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

ISSN 1426–2630

(5) For every set A and for all real bags b_1, b_2 over A such that for every set x such that $x \in A$ holds $b_1(x) \leq b_2(x)$ holds $\sum b_1 \leq \sum b_2$.

(6) For every set A and for all real bags b_1, b_2 over A such that for every set x such that $x \in A$ holds $b_1(x) = b_2(x)$ holds $\sum b_1 = \sum b_2$.

(7) For all sets A_1, A_2 and for every real bag b_1 over A_1 and for every real bag b_2 over A_2 such that $b_1 = b_2$ holds $\sum b_1 = \sum b_2$.

(8) For all sets X, x and for every real bag b over X and for every real number y such that $b = \text{EmptyBag}\, X {+}\!\cdot (x \dot{\longmapsto} y)$ holds $\sum b = y$.

(9) Let X, x be sets, b_1, b_2 be real bags over X, and y be a real number. If $b_2 = b_1 {+}\!\cdot (x \dot{\longmapsto} y)$, then $\sum b_2 = (\sum b_1 + y) - b_1(x)$.

2. Dijkstra's Shortest Path Algorithm: definitions

Let G_1 be a real-weighted w-graph, let G_2 be a w-subgraph of G_1, and let v be a set. We say that G_2 is mincost d-tree rooted at v if and only if the conditions (Def. 1) are satisfied.

(Def. 1)(i) G_2 is tree-like, and

(ii) for every vertex x of G_2 there exists a dpath W_2 of G_2 such that W_2 is walk from v to x and for every dpath W_1 of G_1 such that W_1 is walk from v to x holds W_2.cost() $\leq W_1$.cost().

Let G be a real-weighted w-graph, let W be a dpath of G, and let x, y be sets. We say that W is mincost d-path from x to y if and only if:

(Def. 2) W is walk from x to y and for every dpath W_2 of G such that W_2 is walk from x to y holds W.cost() $\leq W_2$.cost().

Let G be a finite real-weighted w-graph and let x, y be sets. The G .mincost-d-path(x, y) yielding a real number is defined as follows:

(Def. 3)(i) There exists a dpath W of G such that W is mincost d-path from x to y and the G.mincost-d-path(x, y) $=$ W.cost() if there exists a dwalk of G which is walk from x to y,

(ii) the G.mincost-d-path(x, y) $= 0$, otherwise.

Let G be a real-wev wev-graph. The functor DIJK : NextBestEdges(G) yielding a subset of the edges of G is defined by the condition (Def. 4).

(Def. 4) Let e_1 be a set. Then $e_1 \in$ DIJK : NextBestEdges(G) if and only if the following conditions are satisfied:

(i) e_1 joins a vertex from G.labeledV() to a vertex from (the vertices of G) $\setminus$ G.labeledV() in G, and

(ii) for every set e_2 such that e_2 joins a vertex from G.labeledV() to a vertex from (the vertices of G) $\setminus$ G.labeledV() in G holds (the vlabel of G)((the source of G)(e_1)) + (the weight of G)(e_1) $\leq$ (the vlabel of G)((the source of G)(e_2)) + (the weight of G)(e_2).

Let G be a real-wev wev-graph. The functor DIJK : Step(G) yields a real-wev wev-graph and is defined by:

$$\text{(Def. 5)}\quad \text{DIJK : Step}(G) = \begin{cases} G, \text{ if DIJK : NextBestEdges}(G) = \emptyset, \\ (G.\text{labelEdge}(e,1)).\text{labelVertex}((\text{the target of } G)(e), \\ \quad (\text{the vlabel of } G)((\text{the source of } G)(e)) + \\ \quad (\text{the weight of } G)(e)), \text{ otherwise.} \end{cases}$$

Let G be a finite real-wev wev-graph. One can verify that DIJK : Step(G) is finite.

Let G be a nonnegative-weighted real-wev wev-graph. Observe that DIJK : Step(G) is nonnegative-weighted.

Let G be a real-weighted w-graph and let s_1 be a vertex of G. The functor DIJK : Init(G, s_1) yielding a real-wev wev-graph is defined by:

(Def. 6) DIJK : Init$(G, s_1) = G.\text{set}(\text{ELabelSelector}, \emptyset).\text{set}(\text{VLabelSelector}, s_1 \dot{\longmapsto} 0)$.

Let G be a real-weighted w-graph and let s_1 be a vertex of G. The functor DIJK : CompSeq(G, s_1) yielding a real-wev wev-graph sequence is defined as follows:

(Def. 7) DIJK : CompSeq$(G, s_1) \rightarrow 0$ = DIJK : Init(G, s_1) and for every natural number n holds DIJK : CompSeq$(G, s_1) \rightarrow (n+1)$ = DIJK : Step((DIJK : CompSeq$(G, s_1) \rightarrow n$)).

Let G be a finite real-weighted w-graph and let s_1 be a vertex of G. Observe that DIJK : CompSeq(G, s_1) is finite.

Let G be a nonnegative-weighted w-graph and let s_1 be a vertex of G. One can verify that DIJK : CompSeq(G, s_1) is nonnegative-weighted.

Let G be a real-weighted w-graph and let s_1 be a vertex of G. The functor DIJK : SSSP(G, s_1) yields a real-wev wev-graph and is defined by:

(Def. 8) DIJK : SSSP(G, s_1) = (DIJK : CompSeq(G, s_1)).Result().

Let G be a finite real-weighted w-graph and let s_1 be a vertex of G. One can check that DIJK : SSSP(G, s_1) is finite.

3. Dijkstra's Shortest Path Algorithm: theorems

The following propositions are true:

(10) Let G be a finite nonnegative-weighted w-graph, W be a dpath of G, x, y be sets, and m, n be natural numbers. Suppose W is mincost d-path from x to y. Then $W.\text{cut}(m, n)$ is mincost d-path from $(W.\text{cut}(m, n)).\text{first}()$ to $(W.\text{cut}(m, n)).\text{last}()$.

(11) Let G be a finite real-weighted w-graph, W_1, W_2 be dpaths of G, and x, y be sets. Suppose W_1 is mincost d-path from x to y and W_2 is mincost d-path from x to y. Then $W_1.\text{cost}() = W_2.\text{cost}()$.

(12) Let G be a finite real-weighted w-graph, W be a dpath of G, and x, y be sets. Suppose W is mincost d-path from x to y. Then the G.mincost-d-path(x, y) = W.cost().

(13) Let G be a finite real-wev wev-graph. Then

(i) card((DIJK : Step(G)).labeledV()) = card(G.labeledV()) iff DIJK : NextBestEdges(G) = $\emptyset$, and

(ii) card((DIJK : Step(G)).labeledV()) = card(G.labeledV()) + 1 iff DIJK : NextBestEdges(G) $\neq \emptyset$.

(14) For every real-wev wev-graph G holds $G =_G$ DIJK : Step(G) and the weight of G = the weight of DIJK : Step(G) and G.labeledE() $\subseteq$ (DIJK : Step(G)).labeledE() and G.labeledV() $\subseteq$ (DIJK : Step(G)).labeledV().

(15) For every real-weighted w-graph G and for every vertex s_1 of G holds (DIJK : Init(G, s_1)).labeledV() = $\{s_1\}$.

(16) Let G be a real-weighted w-graph, s_1 be a vertex of G, and i, j be natural numbers. If $i \leq j$, then (DIJK : CompSeq(G, s_1)$\rightarrow i$).labeledV() $\subseteq$ (DIJK : CompSeq(G, s_1)$\rightarrow j$).labeledV() and (DIJK : CompSeq(G, s_1)$\rightarrow i$) .labeledE() $\subseteq$ (DIJK : CompSeq(G, s_1)$\rightarrow j$).labeledE().

(17) Let G be a real-weighted w-graph, s_1 be a vertex of G, and n be a natural number. Then $G =_G$ DIJK : CompSeq(G, s_1)$\rightarrow n$ and the weight of G = the weight of DIJK : CompSeq(G, s_1)$\rightarrow n$.

(18) Let G be a finite real-weighted w-graph, s_1 be a vertex of G, and n be a natural number. Then (DIJK : CompSeq(G, s_1)$\rightarrow n$).labeledV() $\subseteq$ G.reachableDFrom(s_1).

(19) Let G be a finite real-weighted w-graph, s_1 be a vertex of G, and n be a natural number.
Then DIJK : NextBestEdges((DIJK : CompSeq(G, s_1)$\rightarrow n$)) = $\emptyset$ if and only if (DIJK : CompSeq(G, s_1)$\rightarrow n$).labeledV() = G.reachableDFrom(s_1).

(20) Let G be a finite real-weighted w-graph, s_1 be a vertex of G, and n be a natural number. Then $\overline{\overline{\text{(DIJK : CompSeq}(G, s_1)\rightarrow n\text{).labeledV()}}}$ = min($n + 1$, card(G.reachableDFrom(s_1))).

(21) Let G be a finite real-weighted w-graph, s_1 be a vertex of G, and n be a natural number. Then (DIJK : CompSeq(G, s_1)$\rightarrow n$).labeledE() $\subseteq$ (DIJK : CompSeq(G, s_1)$\rightarrow n$).edgesBetween((DIJK : CompSeq(G, s_1)$\rightarrow n$) .labeledV()).

(22) Let G be a finite nonnegative-weighted w-graph, s_1 be a vertex of G, n be a natural number, and G_2 be a induced w-subgraph of G, (DIJK : CompSeq(G, s_1)$\rightarrow n$).labeledV(), (DIJK : CompSeq(G, s_1)$\rightarrow n$) .labeledE(). Then

(i) G_2 is mincost d-tree rooted at s_1, and

(ii) for every vertex v of G such that $v \in$ (DIJK : CompSeq$(G, s_1) \to n$) .labeledV() holds the G.mincost-d-path(s_1, v) = (the vlabel of DIJK : CompSeq$(G, s_1) \to n)(v)$.

(23) For every finite real-weighted w-graph G and for every vertex s_1 of G holds DIJK : CompSeq(G, s_1) is halting.

Let G be a finite real-weighted w-graph and let s_1 be a vertex of G. Observe that DIJK : CompSeq(G, s_1) is halting.

One can prove the following three propositions:

(24) For every finite real-weighted w-graph G and for every vertex s_1 of G holds (DIJK : CompSeq(G, s_1)).Lifespan() + 1 = card(G.reachableDFrom(s_1)).

(25) For every finite real-weighted w-graph G and for every vertex s_1 of G holds (DIJK : SSSP(G, s_1)).labeledV() = G.reachableDFrom(s_1).

(26) Let G be a finite nonnegative-weighted w-graph, s_1 be a vertex of G, and G_2 be a induced w-subgraph of G, (DIJK : SSSP(G, s_1)).labeledV(), (DIJK : SSSP(G, s_1)).labeledE(). Then

(i) G_2 is mincost d-tree rooted at s_1, and

(ii) for every vertex v of G such that $v \in G$.reachableDFrom(s_1) holds $v \in$ the vertices of G_2 and the G.mincost-d-path(s_1, v) = (the vlabel of DIJK : SSSP$(G, s_1))(v)$.

4. Prim's Algorithm: preliminaries

The non empty finite subset WGraphSelectors of $\mathbb{N}$ is defined as follows:

(Def. 9) WGraphSelectors = {VertexSelector, EdgeSelector, SourceSelector, TargetSelector, WeightSelector}.

Let G be a w-graph. One can check that G.strict(WGraphSelectors) is graph-like and weighted.

Let G be a w-graph. The functor G.allWSubgraphs() yields a non empty set and is defined as follows:

(Def. 10) For every set x holds $x \in G$.allWSubgraphs() iff there exists a w-subgraph G_2 of G such that $x = G_2$ and $\operatorname{dom} G_2$ = WGraphSelectors.

Let G be a finite w-graph. One can check that G.allWSubgraphs() is finite.

Let G be a w-graph and let X be a non empty subset of G.allWSubgraphs(). We see that the element of X is a w-subgraph of G.

Let G be a finite real-weighted w-graph. The functor G.cost() yields a real number and is defined by:

(Def. 11) $G.\mathrm{cost}() = \sum$ (the weight of G).

The following propositions are true:

(27) For every set x holds $x \in$ WGraphSelectors iff $x =$ VertexSelector or $x =$ EdgeSelector or $x =$ SourceSelector or $x =$ TargetSelector or $x =$ WeightSelector.

(28) For every w-graph G holds WGraphSelectors $\subseteq \operatorname{dom} G$.

(29) For every w-graph G holds $G =_G G$.strict(WGraphSelectors) and the weight of $G =$ the weight of G.strict(WGraphSelectors).

(30) For every w-graph G holds dom(G.strict(WGraphSelectors)) $=$ WGraphSelectors.

(31) For every finite real-weighted w-graph G such that the edges of $G = \emptyset$ holds G.cost() $= 0$.

(32) Let G_1, G_2 be finite real-weighted w-graphs. Suppose the edges of $G_1 =$ the edges of G_2 and the weight of $G_1 =$ the weight of G_2. Then G_1.cost() $=$ G_2.cost().

(33) Let G_1 be a finite real-weighted w-graph, e be a set, and G_2 be a weighted subgraph of G_1 with edge e removed inheriting weight. If $e \in$ the edges of G_1, then G_1.cost() $= G_2$.cost() $+$ (the weight of G_1)(e).

(34) Let G be a finite real-weighted w-graph, V_1 be a non empty subset of the vertices of G, E_1 be a subset of G.edgesBetween(V_1), G_1 be a induced w-subgraph of G, V_1, E_1, e be a set, and G_2 be a induced w-subgraph of G, V_1, $E_1 \cup \{e\}$. If $e \notin E_1$ and $e \in G$.edgesBetween(V_1), then G_1.cost() $+$ (the weight of G)(e) $= G_2$.cost().

5. Prim's Minimum Weight Spanning Tree Algorithm: definitions

Let G be a real-weighted wv-graph. The functor PRIM : NextBestEdges(G) yields a subset of the edges of G and is defined by the condition (Def. 12).

(Def. 12) Let e_1 be a set. Then $e_1 \in$ PRIM : NextBestEdges(G) if and only if the following conditions are satisfied:

(i) e_1 joins a vertex from G.labeledV() and a vertex from (the vertices of G) $\setminus$ G.labeledV() in G, and

(ii) for every set e_2 such that e_2 joins a vertex from G.labeledV() and a vertex from (the vertices of G) $\setminus$ G.labeledV() in G holds (the weight of G)(e_1) $\leq$ (the weight of G)(e_2).

Let G be a real-weighted w-graph. The functor PRIM : Init(G) yields a real-wev wev-graph and is defined by:

(Def. 13) PRIM : Init(G) $=$ G.set(VLabelSelector, choose(the vertices of G) $\dot{\longmapsto}$ 1).set(ELabelSelector, $\emptyset$).

Let G be a real-wev wev-graph. The functor PRIM : Step(G) yielding a real-wev wev-graph is defined by:

(Def. 14) $\text{PRIM} : \text{Step}(G) = \begin{cases} G, \text{ if PRIM} : \text{NextBestEdges}(G) = \emptyset, \\ (G.\text{labelEdge}(e,1)).\text{labelVertex}((\text{the target of } G)(e),1), \text{ if PRIM} : \text{NextBestEdges}(G) \neq \emptyset \text{ and (the source of } G)(e) \in G.\text{labeledV}(), \\ (G.\text{labelEdge}(e,1)).\text{labelVertex}((\text{the source of } G)(e),1), \text{ otherwise.} \end{cases}$

Let G be a real-weighted w-graph. The functor PRIM : CompSeq(G) yields a real-wev wev-graph sequence and is defined by:

(Def. 15) PRIM : CompSeq(G)$\to$0 = PRIM : Init(G) and for every natural number n holds PRIM : CompSeq(G)$\to$($n+1$) = PRIM : Step((PRIM : CompSeq(G)$\to$$n$)).

Let G be a finite real-weighted w-graph. One can check that PRIM : CompSeq(G) is finite.

Let G be a real-weighted w-graph. The functor PRIM : MST(G) yielding a real-wev wev-graph is defined as follows:

(Def. 16) PRIM : MST(G) = (PRIM : CompSeq(G)).Result().

Let G be a finite real-weighted w-graph. Observe that PRIM : MST(G) is finite.

Let G_1 be a finite real-weighted w-graph and let n be a natural number. Observe that every subgraph of G_1 induced by (PRIM : CompSeq(G_1)$\to$$n$).labeledV() is connected.

Let G_1 be a finite real-weighted w-graph and let n be a natural number. Note that every subgraph of G_1 induced by (PRIM : CompSeq(G_1)$\to$$n$).labeledV() and (PRIM : CompSeq(G_1)$\to$$n$).labeledE() is connected.

Let G be a finite connected real-weighted w-graph. Observe that there exists a w-subgraph of G which is spanning and tree-like.

Let G_1 be a finite connected real-weighted w-graph and let G_2 be a spanning tree-like w-subgraph of G_1. We say that G_2 is min-cost if and only if:

(Def. 17) For every spanning tree-like w-subgraph G_3 of G_1 holds G_2.cost() $\leq$ G_3.cost().

Let G_1 be a finite connected real-weighted w-graph. One can check that there exists a spanning tree-like w-subgraph of G_1 which is min-cost.

Let G be a finite connected real-weighted w-graph. A minimum spanning tree of G is a min-cost spanning tree-like w-subgraph of G.

6. Prim's Minimum Weight Spanning Tree Algorithm: theorems

One can prove the following propositions:

(35) Let G_1, G_2 be finite connected real-weighted w-graphs and G_3 be a w-subgraph of G_1. Suppose G_3 is a minimum spanning tree of G_1 and

$G_1 =_G G_2$ and the weight of G_1 = the weight of G_2. Then G_3 is a minimum spanning tree of G_2.

(36) Let G be a finite connected real-weighted w-graph, G_1 be a minimum spanning tree of G, and G_2 be a w-graph. Suppose $G_1 =_G G_2$ and the weight of G_1 = the weight of G_2. Then G_2 is a minimum spanning tree of G.

(37) Let G be a real-weighted w-graph. Then
 (i) $G =_G$ PRIM : Init(G),
 (ii) the weight of G = the weight of PRIM : Init(G),
 (iii) the elabel of PRIM : Init(G) = $\emptyset$, and
 (iv) the vlabel of PRIM : Init(G) = choose(the vertices of G)$\dot{\longmapsto}$1.

(38) For every real-weighted w-graph G holds (PRIM : Init(G)).labeledV() = {choose(the vertices of G)} and (PRIM : Init(G)).labeledE() = $\emptyset$.

(39) For every real-wev wev-graph G such that PRIM : NextBestEdges(G) $\neq \emptyset$ there exists a vertex v of G such that $v \notin G$.labeledV() and PRIM : Step(G) = (G.labelEdge(choose(PRIM : NextBestEdges(G)), 1)).labelVertex(v, 1).

(40) For every real-wev wev-graph G holds $G =_G$ PRIM : Step(G) and the weight of G = the weight of PRIM : Step(G) and G.labeledE() $\subseteq$ (PRIM : Step(G)).labeledE() and G.labeledV() $\subseteq$ (PRIM : Step(G)).labeledV().

(41) Let G be a finite real-weighted w-graph and n be a natural number. Then $G =_G$ PRIM : CompSeq(G)$\rightarrow n$ and the weight of PRIM : CompSeq(G)$\rightarrow n$ = the weight of G.

(42) Let G be a finite real-weighted w-graph and n be a natural number. Then (PRIM : CompSeq(G)$\rightarrow n$).labeledV() is a non empty subset of the vertices of G and (PRIM : CompSeq(G)$\rightarrow n$).labeledE() $\subseteq$ G.edgesBetween((PRIM : CompSeq(G)$\rightarrow n$).labeledV()).

(43) For every finite real-weighted w-graph G_1 and for every natural number n holds every subgraph of G_1 induced by PRIM : CompSeq(G_1)$\rightarrow n$.labeledV() and PRIM : CompSeq(G_1)$\rightarrow n$.labeledE() is connected.

(44) For every finite real-weighted w-graph G_1 and for every natural number n holds every subgraph of G_1 induced by PRIM : CompSeq(G_1)$\rightarrow n$.labeledV() is connected.

(45) For every finite real-weighted w-graph G and for every natural number n holds (PRIM : CompSeq(G)$\rightarrow n$).labeledV() $\subseteq$ G.reachableFrom(choose(the vertices of G)).

(46) Let G be a finite real-weighted w-graph and i, j be natural numbers. If $i \leq j$, then (PRIM : CompSeq(G)$\rightarrow i$).labeledV() $\subseteq$ (PRIM : CompSeq(G)$\rightarrow j$).labeledV() and (PRIM : CompSeq(G)$\rightarrow i$)

.labeledE() $\subseteq$ (PRIM : CompSeq(G)$\rightharpoonup j$).labeledE().

(47) Let G be a finite real-weighted w-graph and n be a natural number. Then PRIM : NextBestEdges((PRIM : CompSeq(G)$\rightharpoonup n$)) $= \emptyset$ if and only if (PRIM : CompSeq(G)$\rightharpoonup n$).labeledV() $=$ G.reachableFrom(choose(the vertices of G)).

(48) Let G be a finite real-weighted w-graph and n be a natural number. Then card((PRIM : CompSeq(G)$\rightharpoonup n$).labeledV()) $=$ min($n +$ 1, card(G.reachableFrom(choose(the vertices of G)))).

(49) For every finite real-weighted w-graph G holds PRIM : CompSeq(G) is halting and (PRIM : CompSeq(G)).Lifespan() $+ 1 =$ card(G.reachableFrom(choose(the vertices of G))).

(50) For every finite real-weighted w-graph G_1 and for every natural number n holds every subgraph of G_1 induced by PRIM : CompSeq(G_1)$\rightharpoonup n$.labeledV() and PRIM : CompSeq(G_1)$\rightharpoonup n$.labeledE() is tree-like.

(51) For every finite connected real-weighted w-graph G holds (PRIM : MST(G)).labeledV() = the vertices of G.

(52) For every finite connected real-weighted w-graph G and for every natural number n holds (PRIM : CompSeq(G)$\rightharpoonup n$).labeledE() $\subseteq$ (PRIM : MST(G)).labeledE().

(53) For every finite connected real-weighted w-graph G_1 holds every induced w-subgraph of G_1, PRIM : MST(G_1).labeledV(), PRIM : MST(G_1).labeledE() is a minimum spanning tree of G_1.

References

[1] Grzegorz Bancerek. Cardinal numbers. *Formalized Mathematics*, 1(**2**):377–382, 1990.
[2] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.
[3] Grzegorz Bancerek. The ordinal numbers. *Formalized Mathematics*, 1(**1**):91–96, 1990.
[4] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.
[5] Czesław Byliński. A classical first order language. *Formalized Mathematics*, 1(**4**):669–676, 1990.
[6] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Formalized Mathematics*, 1(**3**):529–536, 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. Partial functions. *Formalized Mathematics*, 1(**2**):357–367, 1990.
[11] Czesław Byliński. Some basic properties of sets. *Formalized Mathematics*, 1(**1**):47–53, 1990.
[12] Agata Darmochwał. Finite sets. *Formalized Mathematics*, 1(**1**):165–167, 1990.
[13] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[14] Jarosław Kotowicz. Real sequences and basic operations on them. *Formalized Mathematics*, 1(**2**):269–272, 1990.

[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] Gilbert Lee and Piotr Rudnicki. Alternative graph structures. *Formalized Mathematics*, 13(**2**):235–252, 2005.
[19] Piotr Rudnicki. Little Bezout theorem (factor theorem). *Formalized Mathematics*, 12(**1**):49–58, 2004.
[20] Piotr Rudnicki and Andrzej Trybulec. Abian's fixed point theorem. *Formalized Mathematics*, 6(**3**):335–338, 1997.
[21] Piotr Rudnicki and Andrzej Trybulec. Multivariate polynomials with arbitrary number of variables. *Formalized Mathematics*, 9(**1**):95–110, 2001.
[22] Andrzej Trybulec. Subsets of complex numbers. *To appear in Formalized Mathematics.*
[23] Andrzej Trybulec. Domains and their Cartesian products. *Formalized Mathematics*, 1(**1**):115–122, 1990.
[24] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.
[25] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[26] Andrzej Trybulec. Many-sorted sets. *Formalized Mathematics*, 4(**1**):15–22, 1993.
[27] Andrzej Trybulec and Czesław Byliński. Some properties of real numbers. *Formalized Mathematics*, 1(**3**):445–449, 1990.
[28] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[29] Josef Urban. Basic facts about inaccessible and measurable cardinals. *Formalized Mathematics*, 9(**2**):323–329, 2001.
[30] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received February 22, 2005