FORMALIZED MATHEMATICS
Volume 13, Number 2, Pages 305–314
University of Białystok, 2005

Correctnesss of Ford-Fulkerson's Maximum Flow Algorithm[1]

Gilbert Lee[2]
University of Victoria, Victoria, Canada

Summary. We define and prove correctness of Ford-Fulkerson's maximum network flow algorithm at the level of graph manipulations.

MML identifier: `GLIB_005`, version: `7.5.01 4.39.921`

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

1. Preliminary Theorems

Let x be a set and let y be a real number. One can verify that $x \dot{\longmapsto} y$ is real-yielding.

Let x be a set and let y be a natural number. One can verify that $x \longmapsto y$ is natural-yielding.

Let f, g be real-yielding functions. Observe that $f + \cdot g$ is real-yielding.

2. Preliminary Defintions for Ford-Fulkerson Flow Algorithm

Let G be a e-graph. We say that G is complete-elabeled if and only if:

(Def. 1) dom (the elabel of G) = the edges of G.

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

ISSN 1426–2630

Let G be a graph and let X be a many sorted set indexed by the edges of G. Observe that G.set(ELabelSelector, X) is complete-elabeled.

Let G be a graph, let Y be a non empty set, and let X be a function from the edges of G into Y. One can check that G.set(ELabelSelector, X) is complete-elabeled.

Let G_1 be a e-graph sequence. We say that G_1 is complete-elabeled if and only if:

(Def. 2) For every natural number x holds $G_1 \rightarrow x$ is complete-elabeled.

Let G be a w-graph. We say that G is natural-weighted if and only if:

(Def. 3) The weight of G is natural-yielding.

Let G be a e-graph. We say that G is natural-elabeled if and only if:

(Def. 4) The elabel of G is natural-yielding.

Let G_1 be a w-graph sequence. We say that G_1 is natural-weighted if and only if:

(Def. 5) For every natural number x holds $G_1 \rightarrow x$ is natural-weighted.

Let G_1 be a e-graph sequence. We say that G_1 is natural-elabeled if and only if:

(Def. 6) For every natural number x holds $G_1 \rightarrow x$ is natural-elabeled.

One can verify that every w-graph which is natural-weighted is also nonnegative-weighted.

Let us observe that every e-graph which is natural-elabeled is also real-elabeled.

One can verify that there exists a wev-graph which is finite, trivial, tree-like, natural-weighted, natural-elabeled, complete-elabeled, and real-vlabeled.

One can verify that there exists a wev-graph sequence which is finite, natural-weighted, real-wev, natural-elabeled, and complete-elabeled.

Let G_1 be a complete-elabeled e-graph sequence and let x be a natural number. Note that $G_1 \rightarrow x$ is complete-elabeled.

Let G_1 be a natural-elabeled e-graph sequence and let x be a natural number. One can verify that $G_1 \rightarrow x$ is natural-elabeled.

Let G_1 be a natural-weighted w-graph sequence and let x be a natural number. One can verify that $G_1 \rightarrow x$ is natural-weighted.

Let G be a natural-weighted w-graph. One can check that the weight of G is natural-yielding.

Let G be a natural-elabeled e-graph. Note that the elabel of G is natural-yielding.

Let G be a complete-elabeled e-graph. Then the elabel of G is a many sorted set indexed by the edges of G.

Let G be a natural-weighted w-graph and let X be a set. Note that G.set(ELabelSelector, X) is natural-weighted and G.set(VLabelSelector, X) is natural-weighted.

Let G be a graph and let X be a natural-yielding many sorted set indexed by the edges of G. Observe that G.set(ELabelSelector, X) is natural-elabeled.

Let G be a finite real-weighted real-elabeled complete-elabeled we-graph and let s_1, s_2 be sets. We say that G has valid flow from s_1 to s_2 if and only if the conditions (Def. 7) are satisfied.

(Def. 7)(i) s_1 is a vertex of G,

(ii) s_2 is a vertex of G,

(iii) for every set e such that $e \in$ the edges of G holds $0 \leq$ (the elabel of G)(e) and (the elabel of G)(e) $\leq$ (the weight of G)(e), and

(iv) for every vertex v of G such that $v \neq s_1$ and $v \neq s_2$ holds $\sum$((the elabel of G)$\restriction v$.edgesIn()) $= \sum$((the elabel of G)$\restriction v$.edgesOut()).

Let G be a finite real-weighted real-elabeled complete-elabeled we-graph and let s_1, s_2 be sets. Let us assume that G has valid flow from s_1 to s_2. The functor G.flow(s_1, s_2) yields a real number and is defined as follows:

(Def. 8) G.flow(s_1, s_2) $= \sum$((the elabel of G)$\restriction G$.edgesInto($\{s_2\}$)) $- \sum$((the elabel of G)$\restriction G$.edgesOutOf($\{s_2\}$)).

Let G be a finite real-weighted real-elabeled complete-elabeled we-graph and let s_1, s_2 be sets. We say that G has maximum flow from s_1 to s_2 if and only if the conditions (Def. 9) are satisfied.

(Def. 9)(i) G has valid flow from s_1 to s_2, and

(ii) for every finite real-weighted real-elabeled complete-elabeled we-graph G_2 such that $G_2 =_G G$ and the weight of $G =$ the weight of G_2 and G_2 has valid flow from s_1 to s_2 holds G_2.flow(s_1, s_2) $\leq G$.flow(s_1, s_2).

Let G be a real-weighted real-elabeled wev-graph and let e be a set. We say that e is forward labeling in G if and only if the conditions (Def. 10) are satisfied.

(Def. 10)(i) $e \in$ the edges of G,

(ii) (the source of G)(e) $\in G$.labeledV(),

(iii) (the target of G)(e) $\notin G$.labeledV(), and

(iv) (the elabel of G)(e) $<$ (the weight of G)(e).

Let G be a real-elabeled ev-graph and let e be a set. We say that e is backward labeling in G if and only if:

(Def. 11) $e \in$ the edges of G and (the target of G)(e) $\in G$.labeledV() and (the source of G)(e) $\notin G$.labeledV() and $0 <$ (the elabel of G)(e).

Let G be a real-weighted real-elabeled we-graph and let W be a walk of G. We say that W is augmenting if and only if the condition (Def. 12) is satisfied.

(Def. 12) Let n be an odd natural number such that $n < \operatorname{len} W$. Then

(i) if $W(n+1)$ joins $W(n)$ to $W(n+2)$ in G, then (the elabel of G)($W(n+1)$) $<$ (the weight of G)($W(n+1)$), and

(ii) if $W(n+1)$ does not join $W(n)$ to $W(n+2)$ in G, then $0 <$ (the elabel of G)($W(n+1)$).

Let G be a real-weighted real-elabeled we-graph. One can check that every walk of G which is trivial is also augmenting.

Let G be a real-weighted real-elabeled we-graph. Note that there exists a path of G which is vertex-distinct and augmenting.

Let G be a real-weighted real-elabeled we-graph, let W be an augmenting walk of G, and let m, n be natural numbers. Note that $W.\mathrm{cut}(m, n)$ is augmenting.

Next we state two propositions:

(1) Let G_3, G_2 be real-weighted real-elabeled we-graphs, W_1 be a walk of G_3, and W_2 be a walk of G_2. Suppose that

(i) W_1 is augmenting,
(ii) $G_3 =_G G_2$,
(iii) the weight of G_3 = the weight of G_2,
(iv) the elabel of G_3 = the elabel of G_2, and
(v) $W_1 = W_2$.

Then W_2 is augmenting.

(2) Let G be a real-weighted real-elabeled we-graph, W be an augmenting walk of G, and e, v be sets. Suppose that

(i) $v \notin W.\mathrm{vertices}()$, and
(ii) e joins $W.\mathrm{last}()$ to v in G and (the elabel of G)(e) $<$ (the weight of G)(e) or e joins v to $W.\mathrm{last}()$ in G and $0 <$ (the elabel of G)(e).

Then $W.\mathrm{addEdge}(e)$ is augmenting.

3. Algorithm for Finding Augmenting Path in a Graph

Let G be a real-weighted real-elabeled wev-graph. The functor AP : NextBestEdges(G) yielding a subset of the edges of G is defined as follows:

(Def. 13) For every set e holds $e \in$ AP : NextBestEdges(G) iff e is forward labeling in G or backward labeling in G.

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

(Def. 14)
$$\mathrm{AP:Step}(G) = \begin{cases} G, \text{ if AP : NextBestEdges}(G) = \emptyset, \\ G.\mathrm{labelVertex}((\text{the source of } G)(e), e), \\ \quad \text{if AP : NextBestEdges}(G) \neq \emptyset \text{ and (the source of } G) \\ \quad (e) \notin G.\mathrm{labeledV}(), \\ G.\mathrm{labelVertex}((\text{the target of } G)(e), e), \text{ otherwise.} \end{cases}$$

Let G be a finite real-weighted real-elabeled wev-graph. One can check that AP : Step(G) is finite.

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

(Def. 15) AP : CompSeq$(G, s_1)\rightarrow 0$ = G.set(VLabelSelector, $s_1 \dot{\longmapsto} 1$) and for every natural number n holds AP : CompSeq$(G, s_1)\rightarrow(n + 1)$ = AP : Step((AP : CompSeq$(G, s_1)\rightarrow n$)).

Let G be a finite real-weighted real-elabeled we-graph and let s_1 be a vertex of G. One can check that AP : CompSeq(G, s_1) is finite.

The following three propositions are true:

(3) Let G be a real-weighted real-elabeled we-graph and s_1 be a vertex of G. Then

(i) $G =_G$ AP : CompSeq$(G, s_1)\rightarrow 0$,

(ii) the weight of G = the weight of AP : CompSeq$(G, s_1)\rightarrow 0$,

(iii) the elabel of G = the elabel of AP : CompSeq$(G, s_1)\rightarrow 0$, and

(iv) (AP : CompSeq$(G, s_1)\rightarrow 0$).labeledV() = $\{s_1\}$.

(4) Let G be a real-weighted real-elabeled we-graph, s_1 be a vertex of G, and i, j be natural numbers. If $i \leq j$, then (AP : CompSeq$(G, s_1)\rightarrow i$).labeledV() $\subseteq$ (AP : CompSeq$(G, s_1)\rightarrow j$).labeledV().

(5) Let G be a real-weighted real-elabeled we-graph, s_1 be a vertex of G, and n be a natural number. Then $G =_G$ AP : CompSeq$(G, s_1)\rightarrow n$ and the weight of G = the weight of AP : CompSeq$(G, s_1)\rightarrow n$ and the elabel of G = the elabel of AP : CompSeq$(G, s_1)\rightarrow n$.

Let G be a real-weighted real-elabeled we-graph and let s_1 be a vertex of G. The functor AP : FindAugPath(G, s_1) yielding a real-weighted real-elabeled wev-graph is defined as follows:

(Def. 16) AP : FindAugPath(G, s_1) = (AP : CompSeq(G, s_1)).Result().

We now state two propositions:

(6) For every finite real-weighted real-elabeled we-graph G and for every vertex s_1 of G holds AP : CompSeq(G, s_1) is halting.

(7) Let G be a finite real-weighted real-elabeled we-graph, s_1 be a vertex of G, n be a natural number, and v be a set. If $v \in$ (AP : CompSeq$(G, s_1)\rightarrow n$).labeledV(), then (the vlabel of AP : CompSeq$(G, s_1)\rightarrow n$)(v) = (the vlabel of AP : FindAugPath(G, s_1))(v).

Let G be a finite real-weighted real-elabeled we-graph and let s_1, s_2 be vertices of G. The functor AP : GetAugPath(G, s_1, s_2) yielding a vertex-distinct augmenting path of G is defined by:

(Def. 17)(i) AP : GetAugPath(G, s_1, s_2) is walk from s_1 to s_2 and for every even natural number n such that $n \in$ dom AP : GetAugPath(G, s_1, s_2) holds (AP : GetAugPath(G, s_1, s_2))(n) = (the vlabel of AP : FindAugPath(G, s_1))

$((\mathrm{AP}:\mathrm{GetAugPath}(G, s_1, s_2))(n+1))$ if $s_2 \in (\mathrm{AP}:\mathrm{FindAugPath}(G, s_1))$ .labeledV(),

(ii) $\mathrm{AP}:\mathrm{GetAugPath}(G, s_1, s_2) = G.\mathrm{walkOf}(s_1)$, otherwise.

Next we state three propositions:

(8) Let G be a real-weighted real-elabeled we-graph, s_1 be a vertex of G, n be a natural number, and v be a set. Suppose $v \in (\mathrm{AP}:\mathrm{CompSeq}(G, s_1) \rightarrow n)$.labeledV(). Then there exists a path P of G such that P is augmenting and walk from s_1 to v and P.vertices() $\subseteq (\mathrm{AP}:\mathrm{CompSeq}(G, s_1) \rightarrow n)$.labeledV().

(9) Let G be a finite real-weighted real-elabeled we-graph, s_1 be a vertex of G, and v be a set. Then $v \in (\mathrm{AP}:\mathrm{FindAugPath}(G, s_1))$.labeledV() if and only if there exists a path of G which is augmenting and walk from s_1 to v.

(10) Let G be a finite real-weighted real-elabeled we-graph and s_1 be a vertex of G. Then $s_1 \in (\mathrm{AP}:\mathrm{FindAugPath}(G, s_1))$.labeledV() and $G =_G \mathrm{AP}:\mathrm{FindAugPath}(G, s_1)$ and the weight of G = the weight of $\mathrm{AP}:\mathrm{FindAugPath}(G, s_1)$ and the elabel of G = the elabel of $\mathrm{AP}:\mathrm{FindAugPath}(G, s_1)$.

4. Definition of Ford-Fulkerson Maximum Flow Algorithm

Let G be a real-weighted real-elabeled we-graph and let W be an augmenting walk of G. The functor W.flowSeq() yields a finite sequence of elements of $\mathbb{R}$ and is defined by the conditions (Def. 18).

(Def. 18)(i) $\mathrm{dom}(W.\mathrm{flowSeq}()) = \mathrm{dom}(W.\mathrm{edgeSeq}())$, and

(ii) for every natural number n such that $n \in \mathrm{dom}(W.\mathrm{flowSeq}())$ holds if $W(2 \cdot n)$ joins $W(2 \cdot n - 1)$ to $W(2 \cdot n + 1)$ in G, then $W.\mathrm{flowSeq}()(n) =$ (the weight of $G)(W(2 \cdot n))$ − (the elabel of $G)(W(2 \cdot n))$ and if $W(2 \cdot n)$ does not join $W(2 \cdot n - 1)$ to $W(2 \cdot n + 1)$ in G, then $W.\mathrm{flowSeq}()(n) =$ (the elabel of $G)(W(2 \cdot n))$.

Let G be a real-weighted real-elabeled we-graph and let W be an augmenting walk of G. The functor W.tolerance() yielding a real number is defined as follows:

(Def. 19)(i) $W.\mathrm{tolerance}() \in \mathrm{rng}(W.\mathrm{flowSeq}())$ and for every real number k such that $k \in \mathrm{rng}(W.\mathrm{flowSeq}())$ holds $W.\mathrm{tolerance}() \leq k$ if W is non trivial,

(ii) $W.\mathrm{tolerance}() = 0$, otherwise.

Let G be a natural-weighted natural-elabeled we-graph and let W be an augmenting walk of G. Then W.tolerance() is a natural number.

Let G be a real-weighted real-elabeled we-graph and let P be an augmenting path of G. The functor $\mathrm{FF}:\mathrm{PushFlow}(G, P)$ yielding a many sorted set indexed by the edges of G is defined by the conditions (Def. 20).

(Def. 20)(i) For every set e such that $e \in$ the edges of G and $e \notin P$.edges() holds (FF : PushFlow$(G, P))(e) =$ (the elabel of $G)(e)$, and

(ii) for every odd natural number n such that $n < \operatorname{len} P$ holds if $P(n+1)$ joins $P(n)$ to $P(n+2)$ in G, then (FF : PushFlow$(G, P))(P(n+1)) =$ (the elabel of $G)(P(n+1)) + P$.tolerance() and if $P(n+1)$ does not join $P(n)$ to $P(n+2)$ in G, then (FF : PushFlow$(G, P))(P(n+1)) =$ (the elabel of $G)(P(n+1)) - P$.tolerance().

Let G be a real-weighted real-elabeled we-graph and let P be an augmenting path of G. Observe that FF : PushFlow(G, P) is real-yielding.

Let G be a natural-weighted natural-elabeled we-graph and let P be an augmenting path of G. Note that FF : PushFlow(G, P) is natural-yielding.

Let G be a real-weighted real-elabeled we-graph and let P be an augmenting path of G. The functor FF : AugmentPath(G, P) yielding a real-weighted real-elabeled complete-elabeled we-graph is defined as follows:

(Def. 21) FF : AugmentPath$(G, P) = G$.set(ELabelSelector, FF : PushFlow(G, P)).

Let G be a finite real-weighted real-elabeled we-graph and let P be an augmenting path of G. Observe that FF : AugmentPath(G, P) is finite.

Let G be a finite nonnegative-weighted real-elabeled we-graph and let P be an augmenting path of G. Note that FF : AugmentPath(G, P) is nonnegative-weighted.

Let G be a finite natural-weighted natural-elabeled we-graph and let P be an augmenting path of G. Note that FF : AugmentPath(G, P) is natural-weighted and natural-elabeled.

Let G be a finite real-weighted real-elabeled complete-elabeled we-graph and let s_2, s_1 be vertices of G. The functor FF : Step(G, s_1, s_2) yields a finite real-weighted real-elabeled complete-elabeled we-graph and is defined by:

(Def. 22) $$\text{FF : Step}(G, s_1, s_2) = \begin{cases} \text{FF : AugmentPath}(G, \text{AP : GetAugPath}(G, s_1, s_2)), & \text{if } s_2 \in (\text{AP : FindAugPath}(G, s_1)).\text{labeledV}(), \\ G, & \text{otherwise.} \end{cases}$$

Let G be a finite nonnegative-weighted real-elabeled complete-elabeled we-graph and let s_1, s_2 be vertices of G. One can check that FF : Step(G, s_1, s_2) is nonnegative-weighted.

Let G be a finite natural-weighted natural-elabeled complete-elabeled we-graph and let s_1, s_2 be vertices of G. One can verify that FF : Step(G, s_1, s_2) is natural-weighted and natural-elabeled.

Let G be a finite real-weighted w-graph and let s_1, s_2 be vertices of G. The functor FF : CompSeq(G, s_1, s_2) yields a finite real-weighted real-elabeled complete-elabeled we-graph sequence and is defined by the conditions (Def. 23).

(Def. 23)(i) FF : CompSeq$(G, s_1, s_2) {\rightarrow} 0 = G$.set(ELabelSelector, (the edges of $G) \longmapsto 0$), and

(ii) for every natural number n there exist vertices s'_1, s'_2 of FF : CompSeq$(G, s_1, s_2) \rightarrow n$ such that $s'_1 = s_1$ and $s'_2 = s_2$ and FF : CompSeq$(G, s_1, s_2) \rightarrow (n+1) =$ FF : Step(FF : CompSeq$(G, s_1, s_2) \rightarrow n, s'_1, s'_2$).

Let G be a finite nonnegative-weighted w-graph and let s_2, s_1 be vertices of G. One can verify that FF : CompSeq(G, s_1, s_2) is nonnegative-weighted.

Let G be a finite natural-weighted w-graph and let s_2, s_1 be vertices of G. One can check that FF : CompSeq(G, s_1, s_2) is natural-weighted and natural-elabeled.

Let G be a finite real-weighted w-graph and let s_2, s_1 be vertices of G. The functor FF : MaxFlow(G, s_1, s_2) yields a finite real-weighted real-elabeled complete-elabeled we-graph and is defined by:

(Def. 24) FF : MaxFlow(G, s_1, s_2) = (FF : CompSeq(G, s_1, s_2)).Result().

5. Theorems for Ford-Fulkerson Maximum Flow Algorithm

One can prove the following propositions:

(11) Let G be a finite real-weighted real-elabeled complete-elabeled we-graph, s_1, s_2 be sets, and V be a subset of the vertices of G. Suppose G has valid flow from s_1 to s_2 and $s_1 \in V$ and $s_2 \notin V$. Then G.flow$(s_1, s_2) = \sum$((the elabel of G)$\restriction G$.edgesDBetween(V, (the vertices of G) $\setminus V$)) $- \sum$((the elabel of G)$\restriction G$.edgesDBetween((the vertices of G) $\setminus V, V$)).

(12) Let G be a finite real-weighted real-elabeled complete-elabeled we-graph, s_1, s_2 be sets, and V be a subset of the vertices of G. Suppose G has valid flow from s_1 to s_2 and $s_1 \in V$ and $s_2 \notin V$. Then G.flow$(s_1, s_2) \leq \sum$((the weight of G)$\restriction G$.edgesDBetween(V, (the vertices of G) $\setminus V$)).

(13) Let G be a real-weighted real-elabeled we-graph and P be an augmenting path of G. Then $G =_G$ FF : AugmentPath(G, P) and the weight of G = the weight of FF : AugmentPath(G, P).

(14) Let G be a finite real-weighted real-elabeled we-graph and W be an augmenting walk of G. If W is non trivial, then $0 < W$.tolerance().

(15) Let G be a finite real-weighted real-elabeled complete-elabeled we-graph, s_1, s_2 be sets, and P be an augmenting path of G. Suppose $s_1 \neq s_2$ and G has valid flow from s_1 to s_2 and P is walk from s_1 to s_2. Then FF : AugmentPath(G, P) has valid flow from s_1 to s_2.

(16) Let G be a finite real-weighted real-elabeled complete-elabeled we-graph, s_1, s_2 be sets, and P be an augmenting path of G. Suppose $s_1 \neq s_2$ and G has valid flow from s_1 to s_2 and P is walk from s_1 to s_2. Then (G.flow(s_1, s_2)) + P.tolerance() = FF : AugmentPath(G, P).flow(s_1, s_2).

(17) Let G be a finite real-weighted w-graph, s_1, s_2 be vertices of G, and n be a natural number. Then $\mathrm{FF} : \mathrm{CompSeq}(G, s_1, s_2) \rightarrow n =_G G$ and the weight of $G =$ the weight of $\mathrm{FF} : \mathrm{CompSeq}(G, s_1, s_2) \rightarrow n$.

(18) Let G be a finite nonnegative-weighted w-graph, s_1, s_2 be vertices of G, and n be a natural number. If $s_1 \neq s_2$, then $\mathrm{FF} : \mathrm{CompSeq}(G, s_1, s_2) \rightarrow n$ has valid flow from s_1 to s_2.

(19) For every finite natural-weighted w-graph G and for all vertices s_1, s_2 of G such that $s_1 \neq s_2$ holds $\mathrm{FF} : \mathrm{CompSeq}(G, s_1, s_2)$ is halting.

(20) Let G be a finite real-weighted real-elabeled complete-elabeled we-graph and s_1, s_2 be sets such that $s_1 \neq s_2$ and G has valid flow from s_1 to s_2 and there exists no augmenting path of G which is walk from s_1 to s_2. Then G has maximum flow from s_1 to s_2.

(21) Let G be a finite real-weighted w-graph and s_1, s_2 be vertices of G. Then $G =_G \mathrm{FF} : \mathrm{MaxFlow}(G, s_1, s_2)$ and the weight of $G =$ the weight of $\mathrm{FF} : \mathrm{MaxFlow}(G, s_1, s_2)$.

(22) Let G be a finite natural-weighted w-graph and s_1, s_2 be vertices of G. If $s_2 \neq s_1$, then $\mathrm{FF} : \mathrm{MaxFlow}(G, s_1, s_2)$ has maximum flow from s_1 to s_2.

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. 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. Partial functions. *Formalized Mathematics*, 1(**2**):357–367, 1990.

[10] Agata Darmochwał. Finite sets. *Formalized Mathematics*, 1(**1**):165–167, 1990.

[11] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.

[12] Jarosław Kotowicz. Monotone real sequences. Subsequences. *Formalized Mathematics*, 1(**3**):471–475, 1990.

[13] Jarosław Kotowicz. Real sequences and basic operations on them. *Formalized Mathematics*, 1(**2**):269–272, 1990.

[14] Gilbert Lee. Walks in Graphs. *Formalized Mathematics*, 13(**2**):253–269, 2005.

[15] Gilbert Lee. Weighted and Labeled Graphs. *Formalized Mathematics*, 13(**2**):279–293, 2005.

[16] Gilbert Lee and Piotr Rudnicki. Alternative graph structures. *Formalized Mathematics*, 13(**2**):235–252, 2005.

[17] Yatsuka Nakamura, Piotr Rudnicki, Andrzej Trybulec, and Pauline N. Kawamoto. Preliminaries to circuits, I. *Formalized Mathematics*, 5(**2**):167–172, 1996.

[18] Piotr Rudnicki. Little Bezout theorem (factor theorem). *Formalized Mathematics*, 12(**1**):49–58, 2004.

[19] Piotr Rudnicki and Andrzej Trybulec. Abian's fixed point theorem. *Formalized Mathematics*, 6(**3**):335–338, 1997.
[20] Piotr Rudnicki and Andrzej Trybulec. Multivariate polynomials with arbitrary number of variables. *Formalized Mathematics*, 9(**1**):95–110, 2001.
[21] Andrzej Trybulec. Subsets of complex numbers. *To appear in Formalized Mathematics.*
[22] Andrzej Trybulec. Domains and their Cartesian products. *Formalized Mathematics*, 1(**1**):115–122, 1990.
[23] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[24] Andrzej Trybulec. Many-sorted sets. *Formalized Mathematics*, 4(**1**):15–22, 1993.
[25] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[26] Josef Urban. Basic facts about inaccessible and measurable cardinals. *Formalized Mathematics*, 9(**2**):323–329, 2001.
[27] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.
[28] Edmund Woronowicz. Relations defined on sets. *Formalized Mathematics*, 1(**1**):181–186, 1990.

Received February 22, 2005