MIT/LCS/TM-128

MINIMUM REGISTER ALLOCATION IS COMPLETE IN POLYNOMIAL SPACE

Michael C. Loui

March 1979

MINIMUM REGISTER ALLOCATION IS COMPLETE IN POLYNOMIAL SPACE

Michael C. Loui

February 1979

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LABORATORY FOR COMPUTER SCIENCE

CAMBRIDGE MASSACHUSETTS 02139

Minimum Register Allocation is Complete in Polynomial Space

Michael C. Loui†

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts

Abstract. The Minimum Register Allocation Problem is to determine the minimum number of registers required to evaluate an arithmetic expression. A pebble game on directed acyclic graphs is used to prove that this problem is complete in polynomial space.

Key Words: register allocation, pebble game, directed acyclic graph, polynomial space complete, computational complexity.

---

†Supported by the Fannie and John Hertz Foundation.

1. Introduction

The Minimum Register Allocation Problem is a basic problem of computer science [cf. Sethi (1975)]: on a machine with the standard arithmetic operations, how many registers are required to evaluate an arithmetic expression? In this paper we prove that this problem is complete in polynomial space.

We can represent an arithmetic expression E by a directed acyclic graph Γ. The nodes of the graph correspond to subexpressions of E. There are arcs from nodes α and β to node γ when γ is the result of an arithmetic operation applied to α and β. We define a game on Γ to model the evaluation of E with k storage registers. Given k pebbles, one places pebbles on the nodes of Γ in steps according to the following rules:

Pebble Game

(1) A step consists of either
- (a) a placement of a pebble on an empty node, or
- (b) a removal of a pebble from a node, or
- (c) a shift of a pebble to an empty node from one of its immediate predecessors.

(2) A pebble may be placed on or shifted to a node only if there are pebbles on all immediate predecessors of the node. (Thus, a node with no predecessors can be pebbled.)

(3) There are always at most k pebbles on the graph.

The object of the game is to start with no pebbles on Γ and to find a sequence of steps that eventually places a pebble on a designated node of Γ, using at most k pebbles.

Each pebble represents a storage register. Pebbling a node corresponds to storing a value in a register, removing a pebble from a node to releasing a register, and pebbling the designated node to computing the value of the arithmetic expression E. The Minimum Register Allocation Problem is to determine the number of registers required to evaluate E, equivalently, the minimum number of pebbles necessary to pebble the designated node of Γ. This Pebble Game has also been employed to compare flowcharts and recursion schemata [Paterson and Hewitt (1970)] and to study the Turing machine resources time and space [Hopcroft et. al. (1977)].

We can recast the Minimum Register Allocation Problem as a decision problem:

Pebbling Problem. Given a directed acyclic graph Γ and an integer k, can one advance a pebble to a designated node of Γ, starting from an empty configuration, such that at most k pebbles ever appear on Γ?

Sethi [1975] proved that the Pebbling Problem is NP-hard, but the exact complexity of the problem has not been assessed until recently.† Gilbert and Tarjan [1978] and Lingas [1978] showed that more general pebbling problems – for a pebble game on and-or graphs – are polynomial space complete. Earlier, Redziejowski [1969] essentially demonstrated that the Pebbling Problem for *trees* Γ can solved in polynomial time.

Theorem. The Pebbling Problem for directed acyclic graphs with indegree 2 is complete in polynomial space under logarithmic space reduction.

To establish this Theorem, we reduce quantified boolean formulas to the Pebbling Problem in Sections 2 and 3. In Section 4 we modify the construction so that the nodes of the resulting graph have indegree at most 2.

---

†Our proof uses several ideas of J.R. Gilbert, T. Lengauer, and R.E. Tarjan [personal communication], who first obtained the complexity. Nonetheless, this exposition may also be of interest.

2. Construction

Before describing our construction, we adopt a few definitions. A configuration specifies the nodes of a pebbled graph that hold pebbles. (The careful reader may define a configuration to be the set of nodes that hold pebbles; he may then express the following definitions in the terminology of sets.) A step on a configuration is legal if it satisfies restrictions (2) and (3) in the definition in Section 1. A computation that starts from configuration C_0 and ends at configuration C_n is a sequence of configurations $(C_0, C_1, ..., C_n)$ such that for each t, either C_{t-1} is transformed into C_t by a legal step or $C_{t-1} = C_t$. This computation uses k pebbles if in each configuration C_t there at most k pebbles and in some configuration there are k. A computation pebbles a node θ at time t (relative to the start of the computation) if the tth step of the computation places or shifts a pebble onto θ. The notation $[t_1,t_2]$ denotes the interval of times t such that $t_1 \leq t \leq t_2$.

If $S = (C_0, ..., C_n)$ is a computation on Γ and Γ' is a subgraph of Γ, then the restriction of S to Γ' during $[t_1,t_2]$ is the sequence of configurations of pebbles on Γ' in $C_{t_1}, ..., C_{t_2}$. One can confirm routinely that the restriction of a computation to Γ' is itself a computation on Γ'.

Write $\Pi(\theta)$ for the set of immediate predecessors of a node θ.

A node θ_1 is a prerequisite for a node θ_2 if there is a path from θ_1 to θ_2. If θ_1 is a prerequisite for θ_2, and S is a computation that pebbles θ_2 at a time t, and S starts from a configuration in which some path from θ_1 to θ_2 holds no pebbles, then S pebbles θ_1 at some time before t.

Evidently, the Pebbling Problem can be solved in polynomial space: a nondeterministic Turing machine can guess the correct computation, if it exists. To establish completeness, we reduce quantified boolean formulas to this problem. (Stockmeyer [1977] proved that the language of true quantified boolean formulas is complete in polynomial space.) For each quantified boolean formula F we can efficiently generate an acyclic graph such that pebbling a designated node of this graph is tantamount to proving that F is true.

Our construction proceeds inductively on the structure of F. We assume that F has no free variables and has the form

$$Q_r v_r \ldots Q_1 v_1 \,(G), \qquad (*)$$

where each Q_i is either $\forall$ (a universal quantifier) or $\exists$ (an existential quantifier), the variables v_i are distinct, and G is a boolean formula that is a conjunction of s clauses with 3 distinct literals per clause:

$G = H_1 \wedge \ldots \wedge H_s$,

where $H_i = x_i \vee y_i \vee z_i$,

x_i, y_i, z_i are literals, i.e., each is a variable or its negation.

As usual, $\bar{\bar{x}} = x$ for every literal x.

For each subformula f of F with m free variables that contains G as a subformula, we shall construct an acyclic graph $\Gamma(f)$ and define a positive integer $k(f)$. The nodes of $\Gamma(f)$ are divided into two sets $\Phi(f)$, the set of free variable nodes, and $\Theta(f)$, the other nodes. For each free variable w in f there are four nodes $\{\alpha_w, \alpha_{\bar{w}}, \beta_w, \beta_{\bar{w}}\}$ in $\Phi(f)$. Among the nodes of $\Theta(f)$ is the output node ω_f.

A configuration C on $\Gamma(f)$ is an α-β configuration for a variable w if in C either α_w and β_w or $\alpha_{\bar{w}}$ and $\beta_{\bar{w}}$ hold pebbles, but either α_w or $\alpha_{\bar{w}}$ is empty. A configuration is an α-α configuration for w if both α_w and $\alpha_{\bar{w}}$ hold pebbles. By definition, if C is α-α or α-β for w, then there is a pebble on either α_w or $\beta_{\bar{w}}$ (or both) and on either $\alpha_{\bar{w}}$ or β_w.

A configuration on $\Gamma(f)$ is initial if it is α-α or α-β for each free variable and there are no pebbles on $\Theta(f)$. A configuration is strictly initial if it is an initial configuration and for each free variable it is an α-β configuration with just 2 pebbles on $\{\alpha_w, \alpha_{\bar{w}}, \beta_w, \beta_{\bar{w}}\}$: on α_w and β_w, or on $\alpha_{\bar{w}}$ and $\beta_{\bar{w}}$.

A truth assignment is a function from free variables to $\{0,1\}$ (which are interpreted as boolean values for *false* and *true*). A truth assignment $w \mapsto e_w$ to the free variables of f is consistent with a configuration C on $\Gamma(f)$ if in C for each free variable w,

$$e_w = \begin{cases} 1 \text{ if } \beta_w \text{ holds a pebble and } \alpha_{\bar{w}} \text{ does not,} \\ 0 \text{ if } \beta_{\bar{w}} \text{ holds a pebble and } \alpha_w \text{ does not,} \end{cases}$$

It is possible for a configuration to have no consistent truth assignment. If C is an α-α configuration for w, then for consistency e_w can be either 0 or 1. If C is α-α or α-β for every free variable and in C node $\alpha_{\bar{w}}$ is empty, then for consistency e_w must be 1. If C is a strictly initial configuration, then there is only one truth assignment consistent with C.

Call a computation $S = (C_0, ..., C_n)$ on $\Gamma(f)$ dutiful if:

(1) every configuration of S is an α-α or an α-β configuration for each free variable;

(2) the final step of S is a placement or shift onto ω_f, and for all $t < n$, node ω_f is empty in C_t;

(3) S uses at most $2m + k(f)$ pebbles; and

(4) for each literal x, if node α_x is empty in configuration C_u, then it is empty in every C_t for $t \geq u$.

By definition, if $(C_0, ..., C_n)$ is a dutiful computation, then so is $(C_t, ..., C_n)$ for each t. Moreover, by conditions (1) and (4), every truth assignment consistent with C_t is consistent with C_{t-1}; thus, every truth assignment consistent with C_n is consistent with C_0. Consequently, if C_0 is a strictly initial configuration, then the sole truth assignment consistent with C_0 is also the only one consistent with C_n. Condition (1) implies that in every configuration of a dutiful computation there are at least $2m$ pebbles on $\Phi(f)$, hence by condition (3), there are at most $k(f)$ pebbles on $\Theta(f)$.

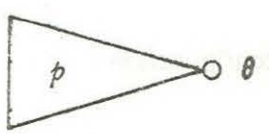
Subformula f with free variables $w_1, ..., w_m$ defines in the usual way a boolean function that we write $f(w_1, ..., w_m)$. We shall establish the following fundamental relationship between f, $k(f)$, and $\Gamma(f)$:

<u>Lemma A.</u> Let F have the form (*). Let f equal G or be a subformula of F of the form $Q_q v_q \dots Q_1 v_1$ (G), $1 \le q \le r$. Let f have free variables $w_1, \dots, w_m$ (where $m = r - q$). Let C_0 be a strictly initial configuration on $\Gamma(f)$ and $w \mapsto e_w$ be the truth assignment consistent with C_0. There is a dutiful computation that starts from C_0 if and only if $f(e_{w_1}, \dots, e_{w_m}) = 1$.

(The free variables $w_1, \dots, w_m$ are necessarily $v_{q+1}, \dots, v_r$.)

The final graph $\Gamma(F)$ is acyclic; one may verify that it can be computed in logarithmic space from F. Since F has no free variables, $\Phi(F) = \varnothing$. Lemma A thus asserts in the case $f = F$ that ω_F can be pebbled by a computation that uses $k(F)$ pebbles on $\Gamma(F)$ if and only if F is true. Thus, together with the construction below, Lemma A implies the Theorem in Section 1.

We present the construction of $\Gamma(f)$ in two stages. At first, we permit nodes with many immediate predecessors. In Section 4 we ensure that each node has indegree at most 2. We sometimes represent a node θ with p immediate predecessors by



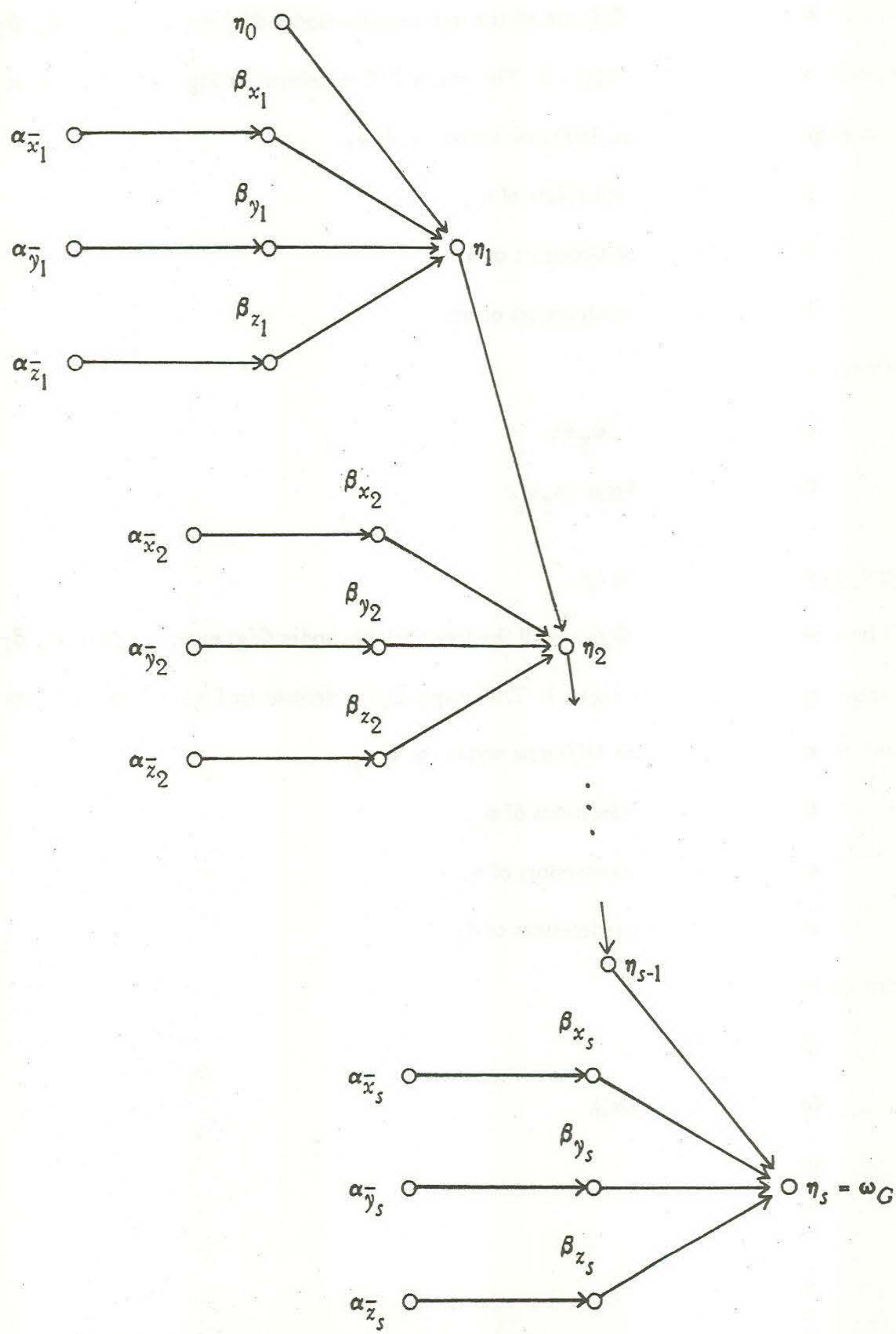
<u>Free Variable Nodes:</u>

In each graph $\Gamma(f)$ there are four nodes $\alpha_w, \alpha_{\bar{w}}, \beta_w, \beta_{\bar{w}}$ and edges $(\alpha_w, \beta_{\bar{w}})$ and $(\alpha_{\bar{w}}, \beta_w)$ for each free variable w in f. The set $\Phi(f)$ consists of these free variable nodes.

<u>Basic Formula:</u> $f = G$.

Set $k(G) = 3$ and $\Gamma(G)$ to be the graph in Figure 1: in addition to the free variable nodes, there is a node η_0, and for each clause $H_i = x_i \vee y_i \vee z_i$ there is a node η_i with edges (β_{x_i}, η_i), (β_{y_i}, η_i), (β_{z_i}, η_i), (η_{i-1}, η_i). Set $\omega_G = \eta_s$.

Figure 1. The graph $\Gamma(G)$.



Universal Quantification: $f = \forall v\ (g)$.

The free variable nodes $\Phi(f)$ are all the free variable nodes $\Phi(g)$ except $\alpha_v, \alpha_{\bar{v}}, \beta_v, \beta_{\bar{v}}$, which become nodes in $\Theta(f)$. Set $k(f) = k(g) + 3$. The graph $\Gamma(f)$ is defined in Figure 2. In addition to the nodes and edges of $\Gamma(g)$ there are $3k(f)$ new nodes: γ, δ, ω_f,

$k(f)$ immediate predecessors of α_v,

$k(f)$-1 immediate predecessors of γ,

$k(f)$-2 immediate predecessors of $\alpha_{\bar{v}}$,

and additional edges:

(α_v,δ), (β_v,δ), (γ,δ), (ω_g,δ),

$(\alpha_{\bar{v}},\omega_f)$, $(\beta_{\bar{v}},\omega_f)$, (δ,ω_f), (ω_g,ω_f).

Existential Quantification: $f = \exists v\ (g)$.

The free variable nodes $\Phi(f)$ are all the free variable nodes $\Phi(g)$ except $\alpha_v, \alpha_{\bar{v}}, \beta_v, \beta_{\bar{v}}$, which become nodes in $\Theta(f)$. Set $k(f) = k(g) + 3$. The graph $\Gamma(f)$ is defined in Figure 3; in addition to the nodes and edges of $\Gamma(g)$ there are $3k(f)$ new nodes: γ, δ, ω_f,

$k(f)$ immediate predecessors of α_v,

$k(f)$-1 immediate predecessors of γ,

$k(f)$-2 immediate predecessors of $\alpha_{\bar{v}}$,

and additional edges:

(β_v,δ), (ω_g,δ), (γ,δ),

$(\alpha_{\bar{v}},\omega_f)$, $(\beta_{\bar{v}},\omega_f)$, (δ,ω_f),

$(\alpha_v,\alpha_{\bar{v}})$.

Figure 2. The graph $\Gamma(f)$ for $f = \forall v\ (g)$.

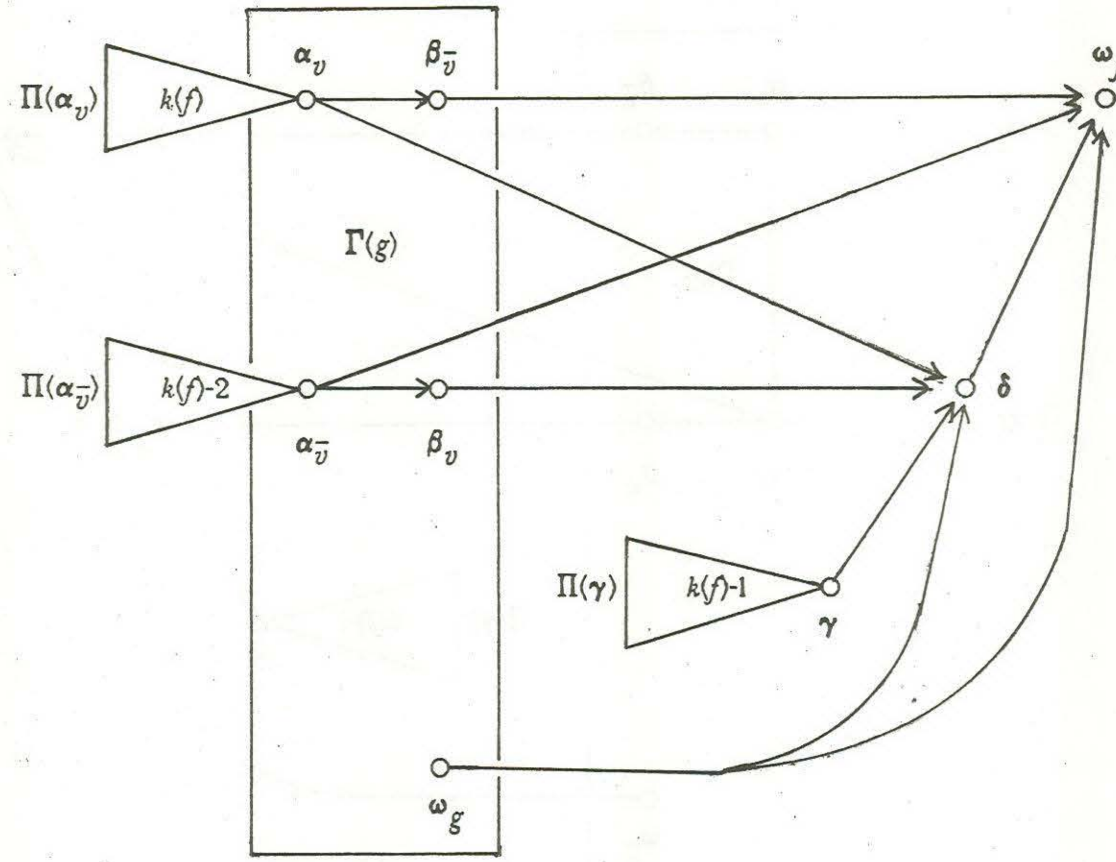
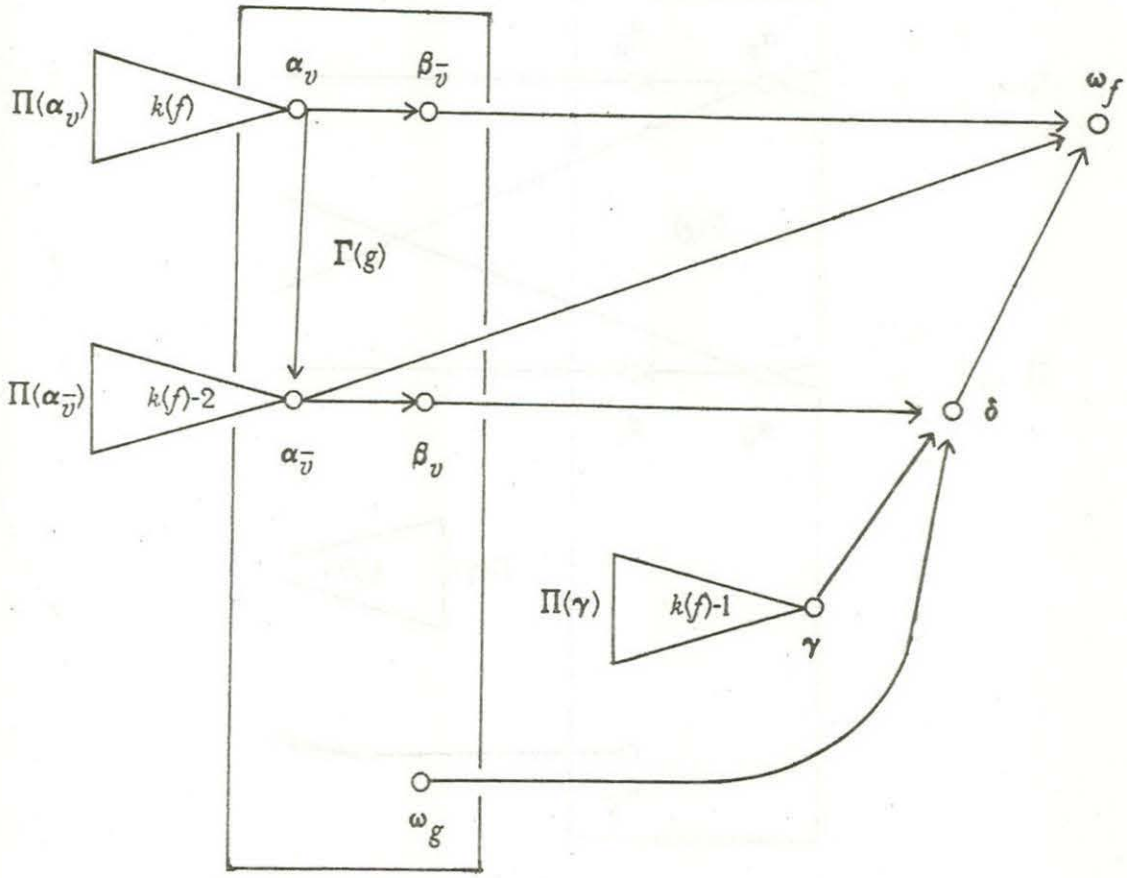


Figure 3. The graph $\Gamma(f)$ for $f = \exists v\ (g)$.



3. Proof of Lemma A

In the proof of Lemma B, we describe a dutiful computation on $\Gamma(f)$ and thereby prove Lemma A in the "if" direction. In Lemma C, we establish the "only if" direction.

Lemma B. Let f have free variables $w_1, ..., w_m$. Let C_0 be a strictly initial configuration on $\Gamma(f)$ and $w \leftrightarrow e_w$ be the truth assignment consistent with C_0. If $f(e_{w_1}, ..., e_{w_m}) = 1$, then there is a dutiful computation that starts from C_0.

Proof. We specify the steps for a dutiful computation by induction on the structure of $\Gamma(f)$.

Case 1. Basic Formula: $f = G$. Refer to Figure 1. Place a pebble on η_0. For each $i = 1, ..., s$, inductively assume that there is a pebble on η_{i-1}. Since $G(e_{w_1}, ..., e_{w_m}) = 1$, $H_i(e_{x_i}, e_{y_i}, e_{z_i}) = 1$, hence e_{x_i}, e_{y_i}, or e_{z_i} is 1, i.e., β_{x_i}, β_{y_i}, or β_{z_i} holds a pebble. Place pebbles on the two other predecessors of η_i, as necessary, and shift the pebble on η_{i-1} to η_i. Remove the zero, one, or two pebbles just placed on β_{x_i}, β_{y_i}, and β_{z_i}. Finally, a pebble reaches $\eta_s = \omega_G$.

Case 2. Universal Quantification: $f(w_1, ..., w_m) = \forall v\, (g(v, w_1, ..., w_m))$. Refer to Figure 2. Since $f(e_{w_1}, ..., e_{w_m}) = 1$, $g(1, e_{w_1}, ..., e_{w_m}) = g(0, e_{w_1}, ..., e_{w_m}) = 1$.

2.1 Place all $k(f)$ pebbles on $\Pi(\alpha_v)$ to pebble α_v. Employ the other $k(f) - 1$ pebbles to pebble γ. Leaving pebbles on α_v and γ, use $k(f) - 2$ pebbles to place a pebble on $\alpha_{\bar{v}}$, and shift it to β_v. Nodes α_v, β_v, and γ now hold pebbles.

2.2 Use the remaining $k(f) - 3 = k(g)$ pebbles to advance a pebble to ω_g via a dutiful computation on $\Gamma(g)$. Nodes α_v, β_v, γ, and ω_g now hold pebbles.

2.3 Shift the pebble on γ to δ.

2.4 Remove all pebbles except those on α_v and δ; use these $k(f) - 2$ pebbles to pebble $\alpha_{\bar{v}}$.

2.5 Shift the pebble on α_v to $\beta_{\bar{v}}$.

2.6 Leaving pebbles on $\alpha_{\bar{v}}$, $\beta_{\bar{v}}$, and δ, use the other $k(f) - 3 = k(g)$ pebbles on $\Theta(g)$ to pebble ω_g again.

2.7 Shift the pebble on δ to ω_f.

Case 3. Existential Quantification: $f(w_1, ..., w_m) = \exists v\ (g(v, w_1, ..., w_m))$. Refer to Figure 2. Put $e = 0$ or 1 so that $g(e, e_{w_1}, ..., e_{w_m}) = 1$.

3.1 Use all $k(f)$ pebbles to pebble α_v. Use the other $k(f) - 1$ pebbles to pebble γ. Leaving pebbles on α_v and γ, pebble $\alpha_{\bar{v}}$ with $k(f) - 2$ pebbles.

3.2 If $e = 0$, shift the pebble on α_v to $\beta_{\bar{v}}$. Otherwise, if $e = 1$, leave the pebble on α_v and shift the pebble on $\alpha_{\bar{v}}$ to β_v.

3.3 Leaving a pebble on γ and pebbles either on α_v and β_v or on $\alpha_{\bar{v}}$ and $\beta_{\bar{v}}$, use the other $k(f) - 3$ pebbles to pebble ω_g via a dutiful computation.

3.4 If $e = 0$, then place a pebble on β_v.

3.5 There are now pebbles on β_v, γ, and ω_g. Shift the pebble on γ to δ.

3.6 If $e = 1$, then remove all pebbles except those on α_v and δ, and use these $k(f) - 2$ pebbles to pebble $\alpha_{\bar{v}}$, then shift the pebble on α_v to $\beta_{\bar{v}}$.

3.7 Nodes $\alpha_{\bar{v}}$, $\beta_{\bar{v}}$, and δ now hold pebbles. Shift one of these to ω_f. □

Lemma C. Let f have free variables $w_1, ..., w_m$. Let C_0 be an initial configuration on $\Gamma(f)$. Let $w \mapsto e_w$ be a truth assignment consistent with a configuration C_n that is α-α or α-β for every free variable of f. If there exists a dutiful computation on $\Gamma(f)$ that starts from C_0 and ends at C_n, then $f(e_{w_1}, ..., e_{w_m}) = 1$.

If C_0 is a strictly initial configuration and a dutiful computation starts from C_0 and ends at configuration C_n, then there is only one truth assignment consistent with C_n, and this is the sole assignment consistent with C_0. Thus, Lemma B and Lemma C together imply Lemma A.

<u>Proof of Lemma C</u>. We proceed by induction on the structure of f.

<u>Case 1</u>. <u>Basic Formula:</u> $f = G$. Refer to Figure 1. Suppose $S = (C_0, \ldots, C_n)$ is a dutiful computation on $\Gamma(G)$. Since S pebbles ω_G, it must pebble every η_i. Recall that the literals x_i, y_i, and z_i are distinct. For each i, pebbles must appear simultaneously on the four nodes β_{x_i}, β_{y_i}, β_{z_i}, η_{i-1} at some time t_i before η_i is pebbled. We claim that one of $\alpha_{\bar{x}_i}$, $\alpha_{\bar{y}_i}$, and $\alpha_{\bar{z}_i}$ is empty in C_{t_i}. If, to the contrary, $\alpha_{\bar{x}_i}$, $\alpha_{\bar{y}_i}$, and $\alpha_{\bar{z}_i}$ all hold pebbles in this configuration, then there are at least $2m + 4$ pebbles on the graph: on $\alpha_{\bar{x}_i}$, $\alpha_{\bar{y}_i}$, $\alpha_{\bar{z}_i}$, β_{x_i}, β_{y_i}, β_{z_i}, η_{i-1}, and at least one on $\{\alpha_x, \beta_{\bar{x}}\}$ for each literal x that does not appear in H_i (because C_{t_i} is α-α or α-β for every variable). Since S is dutiful, it uses at most $2m + 3$ pebbles, and consequently, either $\alpha_{\bar{x}_i}$ or $\alpha_{\bar{y}_i}$ or $\alpha_{\bar{z}_i}$ is empty in C_{t_i}; moreover, this node is empty in every C_t for $t \geq t_i$. Therefore, in C_n, for every i, either $\alpha_{\bar{x}_i}$ or $\alpha_{\bar{y}_i}$ or $\alpha_{\bar{z}_i}$ is empty. It follows that every $H_i(e_{x_i}, e_{y_i}, e_{z_i}) = 1$, and $G(e_{w_1}, \ldots, e_{w_m}) = 1$

<u>Case 2</u>. <u>Universal Quantification:</u> $f(w_1, \ldots, w_m) = \forall v\ (g(v, w_1, \ldots, w_m))$. Refer to Figure 2. Let $S = (C_0, \ldots C_n)$ be a dutiful computation on $\Gamma(f)$. We shall find times $t_1 < t_2 < t_3$ and possibly t_4 at which $k(f)$ pebbles appear on $\Theta(f)$:

C_{t_1}: All $k(f)$ pebbles on $\Pi(\alpha_v)$.

C_{t_2}: A pebble on α_v and $k(f) - 1$ pebbles on $\Pi(\gamma)$.

C_{t_3}: Pebbles on α_v and γ, and $k(f) - 2$ pebbles on $\Pi(\alpha_{\bar{v}})$.

C_{t_4}: Pebbles on α_v or $\beta_{\bar{v}}$, δ, and $k(f) - 2$ pebbles on $\Pi(\alpha_{\bar{v}})$.

Since S is dutiful, it uses at most $k(f)$ pebbles on $\Theta(f)$, and only the last (nth) step pebbles ω_f. Since α_v is a prerequisite for ω_f, S must pebble α_v at some time. Let C_{t_1} be the last configuration in which all $k(f)$ pebbles are on $\Pi(\alpha_v)$. There must be a pebble on α_v or $\beta_{\bar{v}}$ in C_t for all t in $[t_1+1, n-1]$; otherwise, all $k(f)$ pebbles would be required on $\Pi(\alpha_v)$ at some time after t_1 to repebble α_v. Let u_1 be the last time after t_1 that S pebbles δ; for all t in $[u_1, n-1]$, there is a pebble on δ in C_t because δ is a prerequisite for ω_f. Because α_v is an immediate predecessor of δ and $t_1 + 1$ is the last time that S pebbles α_v, for all t in $[t_1+1,u_1-1]$ there is a pebble on α_v in C_t.

Node γ is also a prerequisite for δ. Computation S must pebble γ at some time between t_1 and u_1. Let t_2 be the last time in $[t_1,u_1]$ such that there are $k(f)$ - 1 pebbles on $\Pi(\gamma)$ in configuration C_{t_2}. For all t in $[t_2+1,u_1-1]$ node γ holds a pebble in C_t.

Node β_v is another prerequisite for δ. It must be pebbled at some last time u_2 between t_2 and u_1. For all t in $[u_2,u_1-1]$ node β_v holds a pebble in C_t. Thus, S must pebble the prerequisite $\alpha_{\bar{v}}$ at some time between t_2 and u_2. Let t_3 be the last time in $[t_2,u_2]$ at which $k(f)$ - 2 pebbles appear on $\Pi(\alpha_{\bar{v}})$. For all t in $[t_3+1,u_1-1]$ there is a pebble on either $\alpha_{\bar{v}}$ or β_v in C_t.

Node ω_g is the final prerequisite for δ. It must be pebbled at some first time u_3 between t_3 and u_1.

We claim that the restriction S_g of S to $\Gamma(g)$ during $[t_3+1,u_3]$ is a dutiful computation that starts from an initial configuration on $\Gamma(g)$. Every configuration of the restriction S_g is α-α or α-β for every free variable of $\Gamma(g)$ because there is a pebble on α_v and one on $\alpha_{\bar{v}}$ or β_v during $[t_3+1,u_3]$, and every configuration of S is α-α or α-β for every free variable of f. Computation S_g starts from an initial configuration since no pebbles are on $\Theta(g)$ in C_{t_3+1} (the (t_3+1)th step of S pebbles $\alpha_{\bar{v}}$); it ends with a pebble on ω_g. Subformula g has $m+1$ free variables; since there is a pebble on γ during $[t_3+1,u_3]$, S_g uses at most $2m + k(f) - 1 = 2(m+1) + k(f) - 3 = 2(m+1) + k(g)$ pebbles on $\Gamma(g)$. Finally, if S removes the pebble on $\alpha_{\bar{v}}$ during this interval, then it cannot repebble $\alpha_{\bar{v}}$ because $k(f)$ - 2 pebbles would be required on $\Pi(\alpha_{\bar{v}})$; it follows that if $\alpha_{\bar{v}}$ is empty at some time during S_g, then it remains empty throughout the rest of S_g.

At time u_3, if there are pebbles on both α_v and $\alpha_{\bar{v}}$, then by the inductive hypothesis,

$g(1, e_{w_1}, ..., e_{w_m}) = g(0, e_{w_1}, ..., e_{w_m}) = 1$, and $f(e_{w_1}, ..., e_{w_m}) = 1$.

Otherwise, there is a pebble on α_v, but not on $\alpha_{\bar{v}}$. By induction,

$g(1, e_{w_1}, ..., e_{w_m}) = 1$.

Node $\alpha_{\bar{v}}$ is a prerequisite for ω_f. Let t_4 be the last time (after t_3) at which $k(f)$ - 2 pebbles appear on $\Pi(\alpha_{\bar{v}})$ in C_{t_4}; by definition, $t_4 > u_2$. For all t in $[t_4+1,n-1]$, node $\alpha_{\bar{v}}$ holds a pebble in C_t. In C_{t_4}, since S uses only $k(f)$ pebbles on $\Theta(f)$ and there is a pebble on either α_v or $\beta_{\bar{v}}$, there cannot be pebbles on both β_v and γ. It follows that $t_4 \geq u_1$. There are no pebbles on $\Theta(g)$ in C_{t_4} because there is a pebble on δ and one on α_v or $\beta_{\bar{v}}$.

At some first time u_4 after t_4, node ω_g, a prerequisite of ω_f, must be pebbled. As before, the restriction of S to $\Gamma(g)$ during $[t_4+1,u_4]$ is a dutiful computation that starts from an initial configuration: there are pebbles on $\alpha_{\bar{v}}$, δ, and one on α_v or $\beta_{\bar{v}}$ during this interval. By induction again, $g(0, e_{w_1}, ..., e_{w_m}) = 1$. We conclude that $f(e_{w_1}, ..., e_{w_m}) = \forall v\ (g(v, e_{w_1}, ..., e_{w_m})) = 1$.

Case 3. Existential Quantification: $f(w_1, ..., w_m) = \exists v\ (g(v, w_1, ..., w_m))$. Refer to Figure 3. Let $S = (C_0, ..., C_n)$ be a dutiful computation on $\Gamma(f)$. We shall find times $t_1 < t_2 < t_3$ at which $k(f)$ pebbles appear on $\Theta(f)$:

C_{t_1}: All $k(f)$ pebbles on $\Pi(\alpha_v)$.

C_{t_2}: A pebble on α_v and $k(f)$ - 1 pebbles on $\Pi(\gamma)$.

C_{t_3}: Pebbles on α_v and γ, and $k(f)$ - 2 pebbles on $\Pi(\alpha_{\bar{v}})$.

Since S is dutiful, it uses at most $k(f)$ pebbles on $\Theta(f)$, and only the last (nth) step of S pebbles ω_f.

Since α_v is a prerequisite for ω_f, S must pebble α_v. Let t_1 be the last time at which all $k(f)$ pebbles appear on $\Pi(\alpha_v)$. For all t in $[t_1+1,n-1]$, either α_v or $\beta_{\bar{v}}$ holds a pebble in C_t; otherwise, all $k(f)$ pebbles would be required on $\Pi(\alpha_v)$ at some time after t_1 to pebble α_v. Also, note that S cannot remove the pebble on α_v until it pebbles $\alpha_{\bar{v}}$. Let u_1 be the first time after t_1 at which S pebbles δ, which is a prerequisite for ω_f.

Node γ is a prerequisite for δ, hence S must pebble γ at some time between t_1 and u_1. Let t_2 be the last time between t_1 and u_1 such that $k(f)$ - 1 pebbles appear on $\Pi(\gamma)$. For all t in

$[t_2+1, u_1-1]$ node γ holds a pebble in C_t.

Node $\alpha_{\bar{v}}$ is another prerequisite for δ. Let t_3 be the last time between t_2 and u_1 such that $k(f)$ - 2 pebbles appear on $\Pi(\alpha_{\bar{v}})$. For all t in $[t_3+1, u_1-1]$, either $\alpha_{\bar{v}}$ or β_v holds a pebble; otherwise, $k(f)$ - 2 pebbles would appear on $\Pi(\alpha_{\bar{v}})$ at some time after t_3 to repebble $\alpha_{\bar{v}}$.

Between t_3 and u_1 computation S must pebble ω_g, which is a prerequisite for δ. Let u_2 be the first time after t_3 at which S pebbles ω_g.

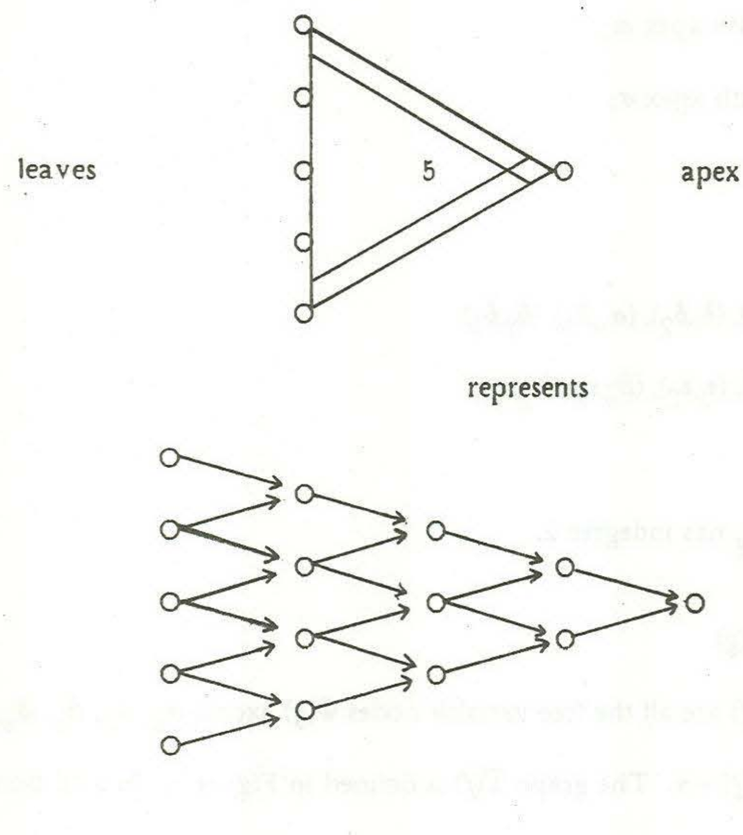
As in Case 2, we can deduce that the restriction of S to $\Gamma(g)$ during $[t_3+1, u_2]$ is a dutiful computation that starts from an initial configuration on $\Gamma(g)$. Nodes α_v and $\alpha_{\bar{v}}$ cannot both become empty during this interval; otherwise, S would eventually be forced to place all $k(f)$ pebbles on $\Pi(\alpha_v)$ to repebble α_v and $\alpha_{\bar{v}}$, which are prerequisites for ω_f. By the inductive hypothesis, $g(e, e_{w_1}, \ldots, e_{w_m}) = 1$ for some e. Ergo, $f(e_{w_1}, \ldots, e_{w_m}) = 1$. □

4. Modification of Construction

We modify the construction of Section 2 to ensure that every node has indegree at most 2. The proofs of Section 3 can be altered in a straightforward fashion to prove that the new $\Gamma(f)$ and the new $k(f)$ satisfy Lemmas B and C.

Figure 4 exhibits a 5-pyramid. A p-pyramid has an apex and p leaves. Cook [1974] showed that pebbling the root of a p-pyramid requires exactly p pebbles. We replace some nodes and their p immediate predecessors with p-pyramids. The new graphs $\Gamma(f)$ for $f = G, f = \forall v\ (g)$, and $f = \exists v\ (g)$ are given in Figures 5, 6, and 7. Each new $\Gamma(f)$ has a node ω_f of indegree 1.

Figure 4. A 5-pyramid.



Basic Formula: $f = G$.

Set $k(G) = 3$ and $\Gamma(G)$ to be the graph in Figure 5: in addition to the free variable nodes, there are two nodes η_0, ω_G and for each clause $H_i = x_i \vee y_i \vee z_i$ there is a 4-pyramid with apex η_i and leaves β_{x_i}, β_{y_i}, β_{z_i}, η_{i-1}. There is also an edge (η_s, ω_G).

Universal Quantification: $f = \forall v\ (g)$.

The free variable nodes $\Phi(f)$ are all the free variable nodes $\Phi(g)$ except α_v, $\alpha_{\bar{v}}$, β_v, $\beta_{\bar{v}}$, which become nodes in $\Theta(f)$. Set $k(f) = k(g) + 4$. The graph $\Gamma(f)$ is defined in Figure 6. In addition to the nodes and edges of $\Gamma(g)$ there are new nodes: δ_1, δ_2, δ_3, ϵ_1, ϵ_2, ϵ_3, ω_f,

a $k(f)$-pyramid with apex α_v,

a $(k(f)$-1)-pyramid with apex γ,

a $(k(f)$-2)-pyramid with apex $\alpha_{\bar{v}}$,

a $(k(f)$-3)-pyramid with apex σ;

and additional edges:

(σ, ω_g),

(β_v, δ_1), (ω_g, δ_1), (γ, δ_2), (δ_1, δ_2), (α_v, δ_3), (δ_2, δ_3),

$(\alpha_{\bar{v}}, \epsilon_1)$, (ω_g, ϵ_1), (δ_3, ϵ_2), (ϵ_1, ϵ_2), $(\beta_{\bar{v}}, \epsilon_3)$, (ϵ_2, ϵ_3),

(ϵ_3, ω_f).

With the new edge (σ, ω_g), node ω_g has indegree 2.

Existential Quantification: $f = \exists v\ (g)$.

The free variable nodes $\Phi(f)$ are all the free variable nodes $\Phi(g)$ except α_v, $\alpha_{\bar{v}}$, β_v, $\beta_{\bar{v}}$, which become nodes in $\Theta(f)$. Set $k(f) = k(g) + 4$. The graph $\Gamma(f)$ is defined in Figure 7. In addition to the nodes and edges of $\Gamma(g)$ there are new nodes: δ_1, δ_2, ϵ_1, ϵ_2, ω_f,

a $k(f)$-pyramid with apex α_v,

a $(k(f)\text{-}1)$-pyramid with apex γ,

a $(k(f)\text{-}2)$-pyramid with apex $\rho_{\bar{v}}$,

a $(k(f)\text{-}3)$-pyramid with apex σ;

and additional edges:

$(\alpha_v,\alpha_{\bar{v}})$, $(\rho_{\bar{v}},\alpha_{\bar{v}})$,

(σ,ω_g),

(β_v,δ_1), (ω_g,δ_1), (γ,δ_2), (δ_1,δ_2),

$(\alpha_{\bar{v}},\epsilon_1)$, (δ_2,ϵ_1), $(\beta_{\bar{v}},\epsilon_2)$, (ϵ_1,ϵ_2),

(ϵ_2,ω_f).

Throughout Sections 2 and 3, any explicit or implicit condition "pebble on $\alpha_{\bar{v}}$" should be replaced by "pebble on $\rho_{\bar{v}}$ or $\alpha_{\bar{v}}$" whenever v is existentially quantified.

Suppose $f = \forall v\ (g)$. Let S be a dutiful computation on $\Gamma(f)$ that starts from an initial configuration, and let u_1 be the last time that S pebbles δ_1. One can show that at some time $t_0 < u_1$ there are pebbles on α_v, on $\alpha_{\bar{v}}$ or β_v, on γ, and $k(f)$ - 3 pebbles on the pyramid whose apex is σ. Let S pebble ω_g at some first time u_2 after t_0; there is a pebble on the pyramid at σ for all t in $[t_0,u_2\text{-}1]$. The restriction of S to $\Gamma(g)$ during $[t_0,u_2]$ is a dutiful computation on $\Gamma(g)$ that starts from an initial configuration. In essence, the pyramid preceding σ forces S into an α-α or α-β configuration for v.

Acknowledgments. Albert Meyer's suggestions improved the precision of the definitions and the clarity of the proofs. Harold Abelson and Jeffrey Jaffe provided valuable comments on a previous draft of the paper. Finally, Robert Tarjan discovered a minor flaw in an earlier version of the modified construction.

Figure 5. The new graph $\Gamma(G)$.

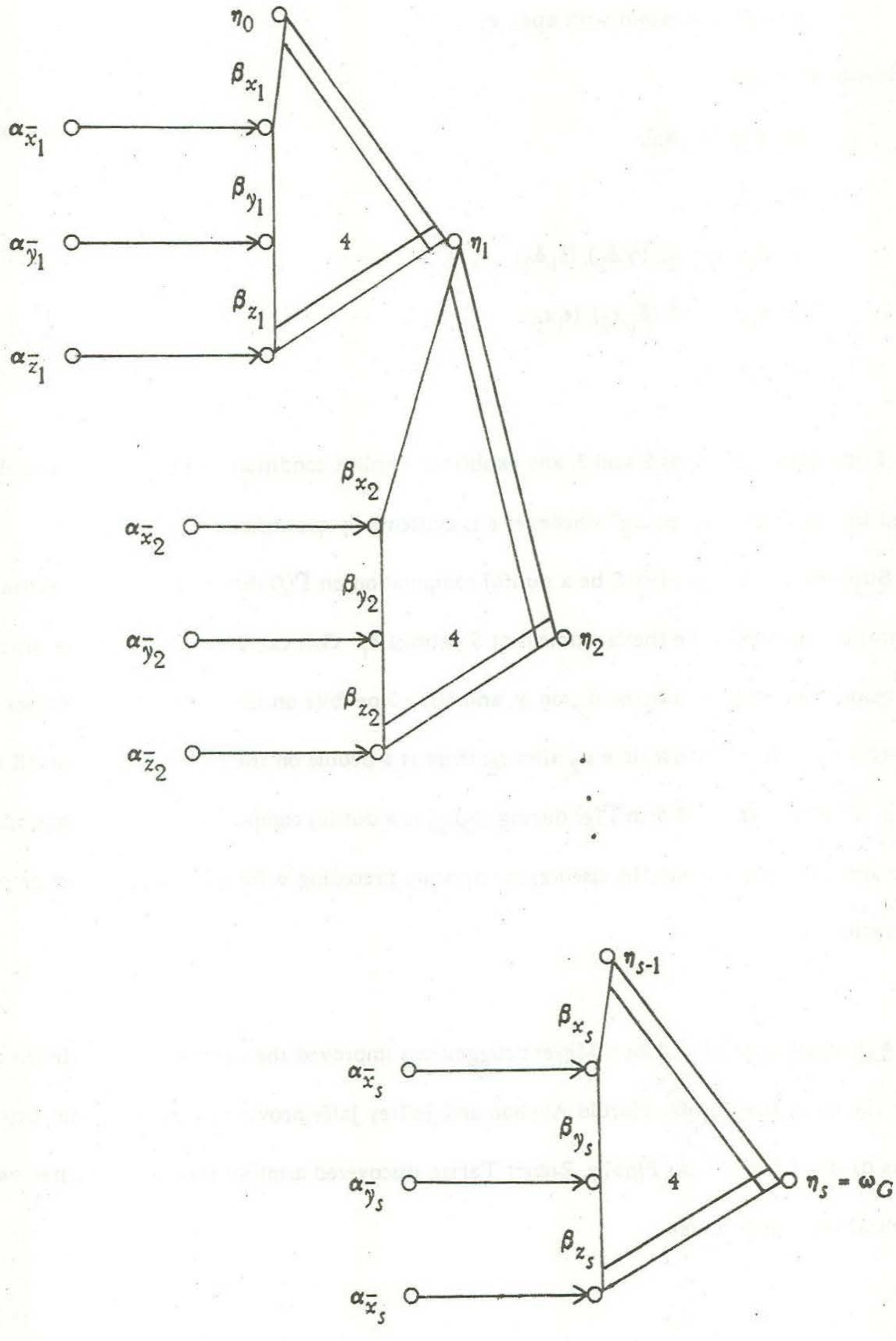


Figure 6. The new graph $\Gamma(f)$ for $f = \forall v\ (g)$.

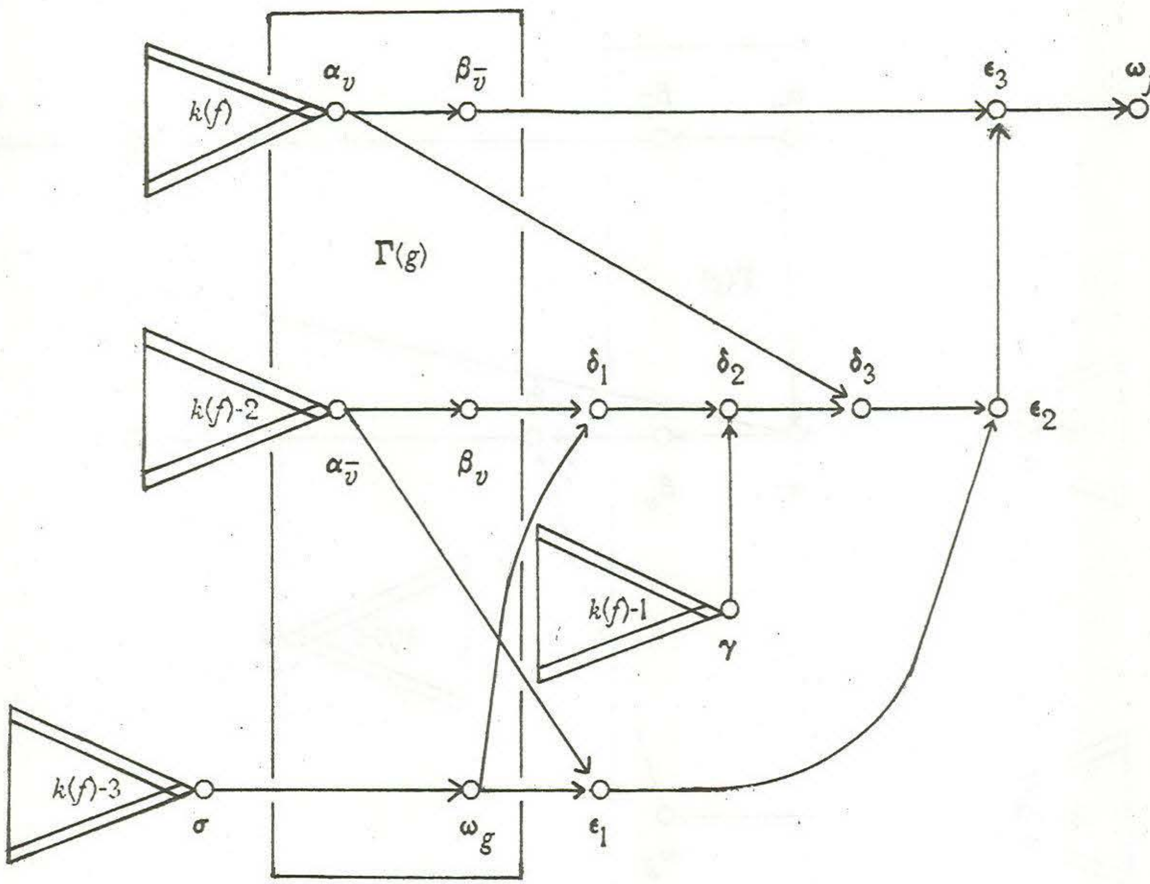
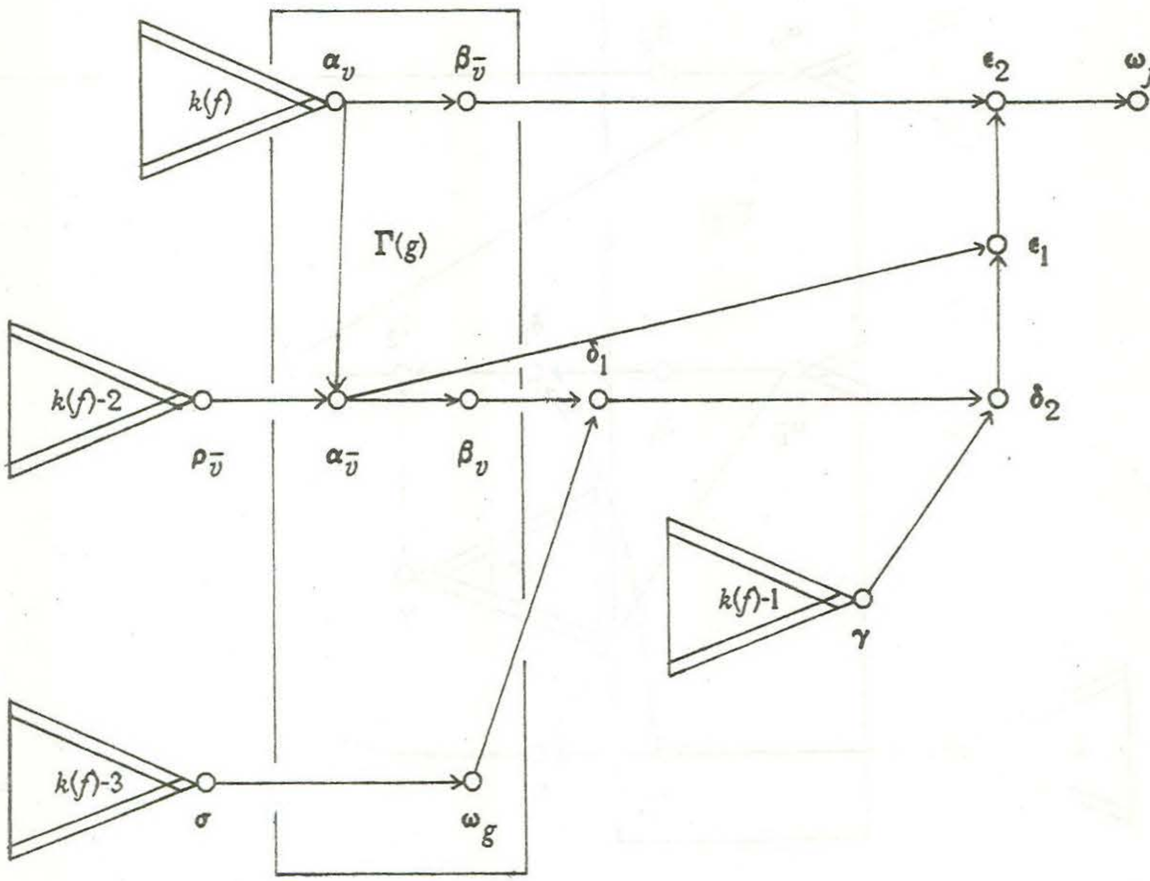


Figure 7. The new graph $\Gamma(f)$ for $f = \exists v\ (g)$.



References

S.A. Cook, "An observation of a time-storage trade-off." *J. Comp. Sys. Sci.* 9 (1974) 308-316.

J. Hopcroft, W. Paul, and L. Valiant, "On time versus space." *J. ACM* 24 (1977) 332-337.

J.R. Gilbert and R.E. Tarjan, "Variations of a pebble game on graphs." Res. Rep. CS78-661, Dept. Comp. Sci., Stanford Univ., Sep. 1978.

A. Lingas, "A PSPACE complete problem related to a pebble game." In *Automata, Languages and Programming* (Fifth Intern. Colloquium, Udine, Italy, July 1978), Lecture Notes in Computer Science vol. 62, ed. G. Ausiello and C. Böhm, Springer-Verlag, Berlin, 1978, pp. 300-321.

M.S. Paterson and C.E. Hewitt, "Comparative schematology." *Rec. Project MAC Conf. on Concurrent Systems and Parallel Computation*, 1970, pp. 119-128.

R.R. Redziejowski, "On arithmetic expressions and trees." *Comm. ACM* 12 (1969) 81-84.

R. Sethi, "Complete register allocation problems." *SIAM J. Comp.* 4 (1975) 226-248.

L.J. Stockmeyer, "The polynomial-time hierarchy." *Theoret. Comp. Sci.* 3 (1977) 1-22.