# On a theorem of McGowan concerning the most recent property of programs

P. Kandzia, H. Langmaack

A 74/07, May 1974

## Introduction

In this paper we discuss the most recent property of ALGOL-like programs ([1],[5],[6]). A program has this property if at run time the static chain pointer of a procedure $\varphi$ always points to the most recent, not yet completed, activation of that procedure $\psi$ which lexicographically encloses $\varphi$. Not all programs of a programming language with procedures have the most recent property; examples are given for BL in [5] and for ALGOL 60 in [2] . So the implementation of block-structured languages cannot be based only on the most recent strategy. In [5] McGowan gives sufficient compile time decidable conditions for the fact that a program $\pi$ has the most recent property. In the following the theorem of McGowan is established on a new basis by investigating the execution tree $T_\pi$ of a program $\pi$.

Some definitions

We deal with the language ALGOL 60-P (P=pure) [4].
ALGOL 60-P has the following main restrictions and modifications compared to ALGOL 60:

a) Only proper procedures, no function procedures are allowed.
b) Specification and value parts in procedure headings are empty.
c) Only identifiers are allowed as actual parameters of procedure statements.
d) Beside begin and end we have an additional pair of statement braces { }. They act as block-begin and block-end. We require that all procedure bodies are included in these braces and we call them body braces in this context. In other contexts they are called call braces.
e) The only data types are real and bool.
f) We exclude arrays, subscripted variables, switches, switch designators, and unsigned integers as labels.

A more complete definition of ALGOL 60-P may be found in [4]. ALGOL 60-P is practically the same languages as BL in McGowan [5].

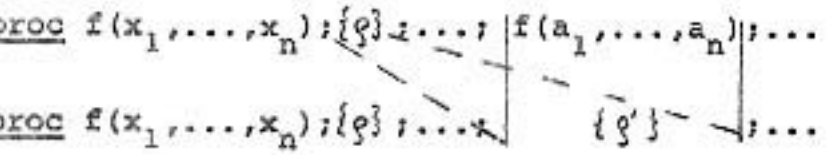
We need some definitions referring to ALGOL 60-P programs. A formal ALGOL 60-P program $\pi$ is a string of basic symbols which can be reduced to the axiom <program> by the formal rules of the context free grammar $\mathcal{G}_P$ for ALGOL 60-P and which has the following property: For every occurence $(i,z)$ of an identifier $z$ in $\pi$ there is exactly one defining occurence $(j,\bar{z}) = \delta(i,z)$ in $\pi$ with $\bar{z} = z$. $i$ and $j$ represent the occurence positions of the identifiers $z$ and $\bar{z}$ in $\pi$. Two formal programs are called identical if they differ only by an admissible renaming of identifiers. In the further text we shall make no difference between identical formal programs.

A formal program is called distinguished if different defining occurences of identifiers $(i,z) \neq (j,\bar{z})$ are denoted by different

identifiers $z \neq \bar{z}$. A formal program is called <u>partially compilable</u> if after replacement of all procedure bodies by empty ones the resulting program $\pi_e$ has the property that, for short, any applied occurence $(i,z)$ of an identifier, bound by the defining occurence $\delta(i,z) = (j,z)$, is applied appropriately according to the definition (it is assumed to be clear what "appropriate application" of identifiers means).

<u>Definition:</u> Let $\pi$ be partially compilable. A formal program $\pi'$ is called to <u>result from $\pi$ by application of the copy rule</u> ($\pi \vdash\!\!- \pi'$) if the following condition holds:
Let $f(a_1,\ldots,a_n)$ be a procedure statement in the main program of $\pi$. Let <u>proc</u> $f(x_1,\ldots,x_n);\{\varsigma\}$ be the associated procedure declaration. Partial compilability guarantees equal number n of actual and formal parameters.
There is a modification of the body $\{\varrho\}$ into a body $\{\varrho'\}$ such that the replacement of $f(a_1,\ldots,a_n)$ in $\pi$ by $\{\varrho'\}$ leads to $\pi'$.

Modification of $\{\varrho\}$ means that all formal parameters $x_i$ occuring in $\varrho$ are replaced by the corresponding actual parameter $a_i$ and that, eventually, identifiers locally defined in $\varrho$ are renamed in admissible manner.
$\{\varrho'\}$ is called <u>generated block</u>; $\{\}$ in $\{\varrho'\}$ are called <u>call braces</u> in contrast to the body braces in $\{\varrho\}$.




If $\pi$ is distinguished, $\pi'$ must not be also distinguished. But we can easily construct an identical and distinguished program $\pi'_d$ if we rename all identifiers local to $\varrho'$ by identifiers which do not yet occur in $\pi'$. In the further text we shall steadily assume that all programs are distinguished and that $\pi'$ with $\pi \vdash\!\!- \pi'$ has been made distinguished like $\pi'_d$. Furtheron, we shall often identify a declaration and the identifier which denotes the declaration.

If $\pi \vdash \pi'$ then all declarations and procedure statements up to $f(a_1,\ldots,a_n)$ are identically copied. Declarations and procedure statements in $\{\rho\}$ have additional copies in $\{\rho'\}$, which we call modified copies. The notion (identical or modified) copy can be easily transferred to $\pi \vdash^{+} \pi'$, $\pi \vdash^{*} \pi'$, where $\vdash^{+}$, $\vdash^{*}$ are the transitive and transitive reflexive resp. closure of $\vdash$ in the set of all formal programs.

Definition: A formal program $\pi$ is called original if $\{\}$ are used only as body braces.

Definition: Let $\pi$ be original; the execution tree $T_\pi$ is the set

$$T_\pi := \{\pi' \mid \pi \vdash^{*} \pi',\ \pi' \text{ has at most one innermost generated block}\}$$

The innermost generated block of $\pi'$ is abbreviated as $IGB(\pi')$.

Let $\pi'' \in T_\pi$, $\pi'' \neq \pi$, $\pi' \vdash \pi''$. In $\pi'$ the calling statement $f'(a_1,\ldots,a_n)$ which is responsible for $IGB(\pi'')$ is uniquely determined; notation: $f'(a_1,\ldots,a_n) = \underline{stat(\pi'')}$. The procedure $f'$ called in $\pi'$ is a copy of exactly one procedure $f$ in $\pi$. We call $f$ the associated procedure $\underline{ass(\pi'')}$ in $\pi$.

An original program $\pi$ is called to have formally correct parameter transmissions if all $\pi'' \in T_\pi$ are partially compilable. A procedure $f$ in an original program $\pi$ is called formally reachable if there is a node $\pi' \in T_\pi$ with $f = ass(\pi')$. If $f$ and $g$ are procedures in an original program $\pi$ then $f$ formally calls $g$ if there are $\pi', \pi'' \in T_\pi$ with $\pi' \vdash^{+} \pi''$ and $ass(\pi') = f$, $ass(\pi'') = g$. $f$ is called formally recursive if $f$ formally calls $f$. It is generally undecidable whether a procedure $f$ in $\pi$ is formally reachable, whether $f$ in $\pi$ is formally recursive and whether $f$ in $\pi$ formally calls another procedure $g$ in $\pi$ [4].

A non-formal identifier $a$ is formally inserted in a formal identifier $x$ :⇔

$(\exists\, \pi'' \in T_\pi \setminus \{\pi\})(stat(\pi'') = f'(\ldots,a_i,\ldots),\ f = ass(\pi'')$ has a declaration $\underline{proc}\ f(\ldots,x_i,\ldots);\{\ldots\}$ in $\pi$ and $x = x_i$, $a_i$ denotes a copy of $a$ in $\pi$.

Algorithm $\mathcal{A}$

We should like to give decidable sufficient conditions when a procedure is formally reachable, when a procedure formally calls another one, when a non-formal identifier is formally inserted in a formal identifier. Let $\pi$ be an original program. Let $p_1,\ldots,p_m$ be the procedure identifiers in $\pi$, let $x_1,\ldots,x_s$ be the formal identifiers in $\pi$, let $p_{m+1},\ldots,p_r$ be the further non-formal identifiers in $\pi$. We consider $\pi$ to be the body of a fictitious procedure named by $P_\pi = p_0$; the main program of $\pi$ is the main part of the body of $P_\pi$ (the main part $mp(p_k)$ of the body of a procedure $p_k$ is that part outside all procedure declarations declared within the body). For the original program $\pi$ we have

$$IGB(\pi) = \pi \ , \ P_\pi = ass(\pi).$$

The above mentioned sufficient conditions are given through an algorithm $\mathcal{A}$ applied to $\pi$. It works with a Boolean insertion matrix I, a Boolean call matrix C and a Boolean reach vektor R initialized in the following way:

| I | $x_1,\ldots,x_s$ |
|---|---|
| $p_1$ | 0 ... 0 |
| ⋮ | ⋮ ⋮ |
| $p_m$ | 0 ... 0 |
| $p_{m+1}$ | 0 ... 0 |
| ⋮ | ⋮ ⋮ |
| $p_r$ | 0 ... 0 |

| C | $p_1 \ldots p_m$ |
|---|---|
| $p_0$ | 0 ...0 |
| $p_1$ | 0 ...0 |
| ⋮ | ⋮ |
| $p_m$ | 0 ...0 |

| R | |
|---|---|
| $p_0$ | 1 |
| $p_1$ | 0 |
| ⋮ | ⋮ |
| $p_m$ | 0 |

We further need a Boolean incorrectness variable V initialized by 0.

Every step S of Algorithm $\mathcal{A}$ has the following partial steps S1,...,S8:

S1: Choose a procedure $p_k$ with $R(p_k) = 1$ and look at $mp(p_k)$.

S2: Let $y_1,\ldots,y_s$ be the (local and global) formal parameters of $p_k$. Choose non-formal identifiers $\alpha_{i_1},\ldots,\alpha_{i_s}$ with $I[\alpha_{i_j},y_j] = 1$.

S3: Imagine $y_1,\ldots,y_s$ to be replaced by $\alpha_{i_1},\ldots,\alpha_{i_s}$ in $mp(p_k)$.

S4: If after this replacement there is an inappropriate identifier application, then let V become 1 and go back to S1; otherwise go on.

S5: Choose a procedure statement $\xi_0(\xi_1,\ldots,\xi_n)$ in $mp(p_k)$. If there is none then go back to S1; otherwise go on.

S6: Look at the procedure statement $\xi_0(\xi_1,\ldots,\xi_n)$ of S5. According to S3 we get $a_0(a_1,\ldots,a_n)$ with

$$\begin{cases} a_r = \xi_r & \text{if } \xi_r \text{ is non-formal} \\ a_r = \alpha_{i_j} & \text{if } \xi_r = y_j \ (y_j \text{ formal}) \end{cases}$$

Appropriate means especially that $a_0$ is a procedure identifier and that n is the number of the formal parameters $z_1,\ldots,z_n$ of $a_0$.

S7: $R[a_0]$ , $C[p_k,a_0]$ and $I[a_1,z_1],\ldots,I[a_n,z_n]$ become 1 .

S8: Return to S1.

Claim: $\alpha$ is nondeterministic and never ends.

To see this we must be sure that every partial step is well executable. This is obvious for every step but the 2nd one, where we need non-formal identifiers $\alpha_{i_1},\ldots,\alpha_{i_s}$ with $I[\alpha_{i_j},y_j] = 1$. That such identifiers are always available is a consequence of the following considerations.

Definition: Let q be an identifier in a program $\pi$. $\varphi(q)$ is defined as the smallest procedure $p \neq q$ which contains the declaration of q. If q is a procedure identifier then $\varphi(q)$ is called the immediate static predecessor of q.

Lemma 1: Immediately before the execution of the first partial step S1 of a step S the following five properties A1 - A5 hold:

A1: $R[p_k] = 1 \succ R[\varphi(p_k)] = 1$

A2: $I[a_{i_j}, y_j] = 1 \succ R[\varphi(a_{i_j})] = 1$

A3: If y is a (local or global) formal parameter of $p_k$ with $R[p_k] = 1$ then there is a non-formal identifier $\alpha_i$ with $I[\alpha_i, y] = 1$.

A4: $C[p_k, p_{k'}] = 1 \succ R[p_k] = R[p_k'] = 1$

A5: $R[p_k] = 1 \not\succ$ there is a sequence $p_o = p_{k_o}, p_{k_1}, \ldots p_{k_\nu} = p_k$, $\nu \geq 0$, with $C[p_{k_i}, p_{k_{i+1}}] = 1$ for $i = 0, \ldots, \nu-1$

Proof: A1 - A5 are trivial before $\mathcal{Q}$ is started. Let A1 - A5 hold immediately before step $S_\mu$, $\mu \geq 1$.

Clear: A1 - A5 hold immediately before step $S_{\mu+1}$ if we return from S4 or S5. Now let us return form S8:

A1: We must show $R[\varphi(a_o)] = 1$ where $a_o$ is taken from the new partial step S6. Let $a_o = \xi_o$, i.e. let $\xi_o$ be non-formal. Then we have

$\varphi(a_o) = p_k$ (where $p_k$ is the procedure just regarded in the new step) or

$\varphi(a_o) = p_{\bar{k}}$ (where $p_{\bar{k}}$ is one of the (immediate or non-immediate) static predecessors of $p_k$).

By induction hypothesis A1 we have $R[\varphi(a_o)] = 1$.

Let $\xi_o$ be formal with $I[a_o, \xi_o] = 1$. Then by induction hypothesis A2 we have $R[\varphi(a_o)] = 1$.

A2: We must show $R[\varphi(a_i)] = 1$, $i > 0$. We argue as above.

A3: We must show: For every (local or global) formal parameter y of procedure $a_o$ there is a non-formal identifier $\alpha_i$ with $I[\alpha_i,y] = 1$. Let y be local and $y = z_j$. Then by S7 $I[a_j,y] = I[a_j,z_j] = 1$. Let y be global. Then y is a (local or global) formal parameter of $\varphi(a_o)$. Because of A1 above $R[\varphi(a_o)] = 1$ holds. By induction hypothesis A3 we have a non-formal identifier $\alpha_i$ with $I[\alpha_i,y] = 1$.

A4 and A5 are easy to show. Q.E.D.

Every not ending sequence $\gamma$ of steps $S_1,S_2,S_3,\ldots$ of Algorithm $\mathcal{A}$ has a first step $S_t$ such that $S_t,S_{t+1},S_{t+2},\ldots$ do not change the values of I,C,R or V. We call these finally constant matrices $I^\gamma$, $C^\gamma$, vector $R^\gamma$, and variable $V^\gamma$ the result of $\mathcal{A}$ with respect to sequence $\gamma = S_1,S_2,S_3,\ldots$. Different step sequences $\gamma,\gamma'$ may yield different results, but there is always another sequence $\gamma''$ with $I^\gamma \cup I^{\gamma'} \leq I^{\gamma''}$, $C^\gamma \cup C^{\gamma'} \leq C^{\gamma''}$, $R^\gamma \cup R^{\gamma'} \leq R^{\gamma''}$, $V^\gamma \cup V^{\gamma'} \leq V^{\gamma''}$. The unions $I_\pi := \bigcup_\gamma I^\gamma$, $C_\pi := \bigcup_\gamma C^\gamma$, $R_\pi := \bigcup_\gamma R^\gamma$, $V_\pi := \bigcup_\gamma V^\gamma$ are called the result of $\mathcal{A}$. The result can effectively be determined if we go systematically through all choices which are offered in the partial steps S1, S2 and S5.

Some properties of programs basing on Algorithm $\mathcal{A}$

By means of Algorithm $\mathcal{A}$ we define some (compile time decidable) properties of a program $\pi$:

We say that $\pi$ has potentially correct parameter transmissions if $V_\pi = 0$. We call a procedure $p_k$ potentially reachable if $R_\pi[p_k] = 1$. We say a procedure $p_k$ potentially calls procedure $p_{k'}$ if $C^+_\pi[p_k,p_{k'}] = 1$, where $C^+$ is the transitive closure of C. A procedure $p_k$ is called potentially recursive if $p_k$ potentially calls $p_k$. A non-formal identifier $p_i$ is called to be potentially inserted in the formal identifier $x_j$ if $I_\pi[p_i,x_j] = 1$.

We may prove

Theorem 1: 1. If $\pi$ has potentially correct parameter transmissions then $\pi$ has formally correct parameter transmissions, too.
2. If a procedure g in $\pi$ is formally reachable then g is potentially reachable.
3. If a procedure g in $\pi$ formally calls a procedure f in $\pi$ then g potentially calls f.
4. If a procedure g in $\pi$ is formally recursive then g is potentially recursive.
5. If a non-formal identifier p is formally inserted in the formal identifier x then p is potentially inserted in x.

Proof: Let $\pi'$ be a program in $T_\pi$ and $g = ass(\pi')$.
The theorem is a consequence of the following claims:

Claim 1: g is potentially reachable.

Claim 2: If $\pi'$ is not partially compilable then $\pi$ has not potentially correct parameter transmissions.
Let $\pi'$ be partially compilable and let $f'(a'_1,\ldots,a'_n)$ be a procedure statement of IGB($\pi'$) in the main program of $\pi'$. Let f' denote a copy of f with the declaration

$$\underline{\text{proc}}\ f(x_1,\ldots,x_n)\ ;\ \{\varrho\}$$

in $\pi$ and let $a'_i$ denote copies of identifiers $a_i$ in $\pi$.

Claim 3: g potentially calls f

Claim 4: $a_i$ is potentially inserted in $x_i$.

The claims are proved by induction. They are obvious for $\pi' = \pi$. Now let $\pi \overset{*}{\vdash} \pi'' \vdash \pi'$ in $T_\pi$ and let Claim 1,...,4 be fulfilled for $\pi''$ and its (eventual) predecessors.

Claim 1 for $\pi'$: Because of claim 3 for $\pi''$ we have $C_{\pi''}[\ldots,g] = 1$ and due to property A4 $R_{\pi''}[g] = 1$.

Claim 2 for $\pi'$: Because of claim 1 for $\pi'$, claim 4 for $\pi^*$ and its predecessors, and because $\pi'$ is not partially compilable Algorithm $\mathcal{A}$ yields $V_\pi = 1$.
Claim 3 and 4 for $\pi'$: Because of claim 2 for $\pi'$, claim 4 for $\pi^*$ and its predecessors, and because $\pi'$ is partially compilable Algorithm $\mathcal{A}$ yields $C_\pi[g,f] = 1$ and $I_\pi[a_i,x_i] = 1$. Q.E.D.

A new approach to the theorem of McGowan

We want to represent systematically the modification of bodies $\{\varrho\}$, such that in a program $\pi'' \in T_\pi$ it is recognizable in which program $\pi'$ with $\pi \overset{*}{\vdash} \pi' \overset{*}{\vdash} \pi''$ the renaming of a certain identifier has happened. The programs $\pi'' \in T_\pi$ will be characterized by strings $\in \mathbb{N}^*$. For this purpose we look at the statical procedure structure of a body which has the form

| | | |
|---|---|---|
| $\left\{\begin{matrix}\\ \text{(or } \underline{\text{begin}}\text{)}\end{matrix}\right.$ | $\ldots \underline{\text{proc}}\ p_1(x_{11},\ldots,x_{1,n_1});\{\ldots\}$ | $0 \leq n_1$ |
| | $\vdots$ | |
| | $\underline{\text{proc}}\ p_m(x_{m1},\ldots,x_{m,n_m});\{\ldots\}$ | $0 \leq n_m, m \geq 0$ |
| | $\vdots$ | |
| | $a_{10}(a_{11},\ldots,a_{1,r_1})$ | $0 \leq r_1$ |
| | $\vdots$ | |
| | $a_{q0}(a_{q1},\ldots,a_{q,r_q})\quad \ldots\ \left\}\begin{matrix}\\ \text{(or } \underline{\text{end}}\text{)}\end{matrix}\right.$ | $0 \leq r_q, q \geq 0$ |

if we neglect all declarations and statements which are not procedure declarations and procedure statements resp. Block nesting within bodies is neglected too.

We define:

The relative position of the declaration of a procedure $p_i$: $P(p_i) := i$;
the relative position of a statement $\Sigma = a_{i0}(\ldots)$: $S(a_{i0}(\ldots)) := i$;
the relative position of a formal parameter $x = x_{jk}$: $X(x_{jk}) := k$.

P,S,X are extended to all programs $\pi' \in T_\pi$ which is demonstrated only for S:

$S(\Sigma') := S(\Sigma)$ where statement $\Sigma'$ in $\pi'$ is a copy of statement $\Sigma$ in $\pi$.

The above mentioned characterization of programs $\pi'' \in T_\pi$ can be stated in the following way:

$$\tau(\pi'') := \begin{cases} \varepsilon \quad \text{(empty string)} & \text{if } \pi'' = \pi \\ \tau(\pi')\ \S(\text{stat}(\pi'')) & \text{if } \pi' \vdash\!\!- \pi'' \end{cases}$$

$\tau$ is a 1-1-mapping so that all programs $\pi' \in T_\pi$ could be identified with their strings $\tau(\pi')$. $\tau(T_\pi)$ is an (eventually infinite) tree T with the following characterizing properties:

1.) $T \subseteq \mathbb{N}^*$ , $T \neq \emptyset$

2.) T is closed under initial segment relation
i.e. if $st \in T$ then $s \in T$, too.

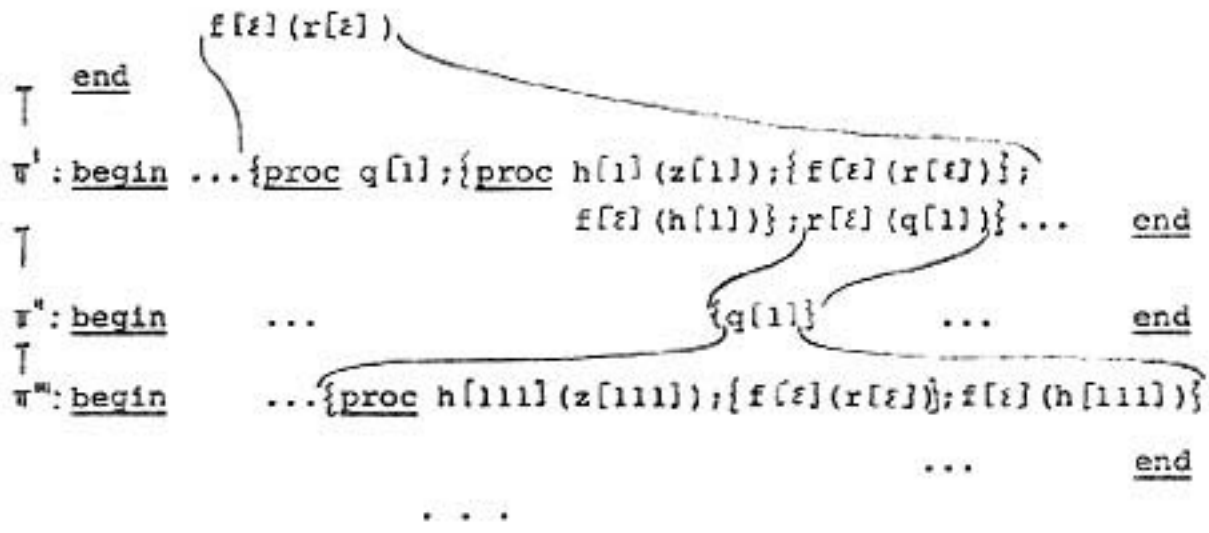
3.) If $t\nu$ with $\nu > 1$ is in T then $T(\nu-1)$ is so , too.

4.) Any node t in T has at most finitely many immediate successors $t\nu$ in T with $\nu \in \mathbb{N}$.

The systematic renaming of identifiers in applying the copy rule can now be established: Identifiers in a program $\pi' \in T_\pi$ will be pairs (id,t), written id[t] , where id is an identifier of old kind in $\pi$ and t is an element of $\tau(T_\pi) \subseteq \mathbb{N}^*$. All identifiers in $\pi$ are indexed by $\varepsilon = \tau(\pi)$: id[ε] . Let $\pi' \vdash\!\!- \pi''$ in $T_\pi$ . All identifiers local to IGB($\pi''$) get the index $\tau(\pi'')$, while the remaining identifiers get the same index as in $\pi'$.

Example 1:

```
π1: begin proc r[ε](y[ε]);{y[ε]};
          proc f[ε](x[ε]);{proc q[ε];{proc h[ε](z[ε]);{f[ε](r[ε])};
                                       f[ε](h[ε])}; x[ε](q[ε])};
```




With the new systematic renaming of identifiers in programs $\pi'' \in T_\pi$ we can define some terms important for the definition and investigation of the most recent property.

Definition: Let $\pi'' \in T_\Gamma \setminus \{\sigma\}$ ;

the dynamic predecessor of $\pi''$ is $dyn(\pi'') := \pi'$ with $\pi' \vdash \pi''$;

the dynamic chain of $\pi''$ is the set $\vartheta(\pi'') := \{dyn^\nu(\pi'') \mid \nu = 0,1,\ldots, |\tau(\pi'')|\}$

( $dyn^0$ is the identical mapping).

The static predecessor of $\pi''$ is defined $st(\pi'') := \tau^{-1}(t_0)$

where $stat(\pi'') = f[t_0](a_1[t_1],\ldots,a_n[t_n])$

The set $\vartheta(\pi'') := \{st^\nu(\pi'') \mid \nu = 0,\ldots, \lambda(ass(\pi''))$ is called the static chain of $\pi''$. ($st^0$ is the identical mapping). $\lambda(ass(\pi''))$ is the static level of $ass(\pi'')$ which is introduced in the following definition.

Definition: The static level of an original program $\tau$ (which is regarded as body of a procedure $P_\tau$ ) is $\lambda(P_\pi) := 0$;
static level of a procedure $p \neq P_\tau$: $\lambda(p) := \lambda(\vartheta(p)) + 1$.

We want to deduce theorems about the most recent property. This property has to do with the most recent, not yet completed, activations of those procedures which lexicographically enclose

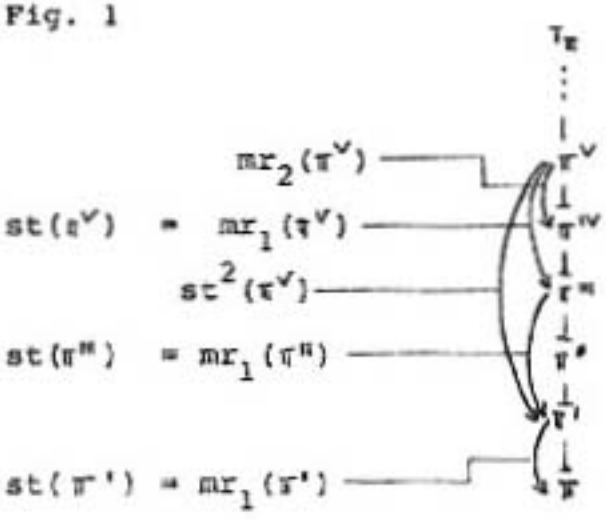
$ass(\pi")$, $\pi" \in T_\pi$. We need suitable definitions.

Definition: Let $\pi" \in T_\pi \setminus \{\pi\}$ and $0 \leq \mu \leq \lambda(ass(\pi"))$. The $\mu$-th most recent predecessor of $\pi"$ is defined as $mr_\mu(\pi") := dyn^\nu(\pi")$, where $\nu$ is the smallest number such that $ass(dyn^\nu(\pi")) = ass\,(st^\mu(\pi"))$. $mr_0$ is the identical mapping.

dyn, st and $mr_\mu$ are functions from $T_\pi \setminus \{\pi\}$ to $T_\pi$.
If $\mu_1 + \ldots + \mu_r = \lambda(f)$, $r \geq 0$, then $mr_{\mu_1} \circ \ldots \circ mr_{\mu_r}(\pi") = \pi$.

As an example we may treat the last program $\pi_1$. In fig. 1 we have a part of the diagram of $T_\pi$ with some arrows representing the functions st, $st^2$, $mr_1$, $mr_2$. Beside this we have listed the elements $stat(\pi^i)$ and $ass(\pi^i)$, $\pi^i \in \{\pi_1, \pi', \pi", \pi"', \pi^{IV}, \pi^V\}$, $(stat(\pi) = P_\pi$ , $ass(\pi) = P_\pi)$.

Fig. 1




| $stat(\pi^i)$ | $ass(\pi^i)$ |
|---|---|
| $h[111](q[1111])$ | h |
| $f[\varepsilon](h[111])$ | f |
| $q[1]$ | q |
| $r[\varepsilon](q[1])$ | r |
| $f[\varepsilon](r[\varepsilon])$ | f |
| $P_\pi$ | $P_\pi$ |

Definition: Let $\pi" \in T_\pi \setminus \{\pi\}$. $\pi"$ is called to fulfill the weak (strong) most recent property if $st(\pi") = mr_1(\pi")$ (resp. $st^\mu(\pi") = mr_\mu(\pi")$ for all $\mu$ with $0 \leq \mu \leq \lambda(ass(\pi"))$).

The original program $\pi$ is called to fulfill the weak (strong) most recent property, if all $\pi" \in T_\pi \setminus \{\pi\}$ fulfill the respective property.

We are interested in a necessary condition when $\pi$ does not fulfill the strong most recent property. Then there is a program $\pi'' \neq \pi$ in $T_\pi$ and a number $\mu > 0$ such that

a) $st^{\mu}(\pi'') \neq mr_{\mu}(\pi'')$

and b) $st^{m}(\pi') = mr_{m}(\pi')$ for all $\pi' \vdash^{+} \pi''$, $\pi' \neq \pi$, and all numbers $m$, $0 \leq m \leq \lambda(ass(\pi'))$.

Before investigating the consequences of a) and b) we need some minor considerations on non-formal identifiers in a program $\pi'' \in T_\pi \setminus \{\pi\}$. We must distinguish exactly between non-formal identifiers on "non-formal position" and non-formal identifiers on "formal position". An occurence of a $[t]$ in $\pi''$ is on "non-formal position", if it corresponds to (is on the same place as) the occurence of the non-formal identifier $a[\xi]$ in $\pi$. $a[t]$ is on "formal position", if it is on the place of the formal identifier $x[\xi]$ in $\pi$.

How does $a[t]$ come to the position of the formal identifier $x[\xi]$ ? Let the declaring occurence of $x = x[\xi]$ be in

proc g(...,x,...);{...};

$a[t]$ comes to the position of x by the calling statement

$g[t_o](...,a[t],...)$

in $\pi^*$ with $\pi^* \vdash^{+} \pi''$. The corresponding call in $\pi$ is

$\alpha$) $g[\xi](...,a[\xi],...)$ or

$\beta$) $g[\xi](...,y[\xi],...)$, where $y[\xi]$ is formal.

In case $\beta$) $a[t]$ is already in an enclosing block on formal position (this time of $y[\xi]$). The consideration continues in that way until we have case $\alpha$).

For case $\alpha$) and case $\beta$), we show two lemmata:

Lemma 2 (case $\alpha$): Let $\pi'' \in T_\pi$ and $a[t]$ an occurence of a non-formal identifier in IGB($\pi''$); let $a[t]$ stand on the place of $a[\xi]$ in the body of $p = ass(\pi'')$ in $\pi$.

Then

$$\tau^{-1}(t) = st^{\bar{d}}(\pi'') \quad \text{with } d = \lambda(p) - \lambda(\varphi(a))$$

$$\bar{d} = \begin{cases} d & \text{if } d \geq 0 \\ 0 & \text{if } d < 0 \end{cases}$$

especially: $\delta(\tau^{-1}(t)) \leq \delta(\pi'')$.

Proof: Trivial for $\pi'' = \pi$ .

Let $\pi' \vdash\!\!- \pi''$ by the statement $p[t_o](...)$ and the lemma hold for $\pi'$ and its dynamic predecessors. If a is local to the body of p then $d \leq 0$ and $\tau^{-1}(t) = \pi'' = st^{\bar{d}}(\pi'')$ by construction of indexing.

If a is global then $d > 0$ and $a[t]$ occurs in the body of the procedure $p[t_o]$ called. $a[t]$ occurs in $IGB(\tau^{-1}(t_o))$ ; $d-1 \geq 0$ and by induction hypothesis

$$\tau^{-1}(t) = st^{d-1}(\tau^{-1}(t_o)) = st^{d-1}(st(\pi'')) = st^{d}(\pi'')$$

Lemma 3( case $\beta$ ): Let $\pi'' \in T_\pi$ and $a[t]$ an occurence of a non-formal identifier in $IGB(\pi'')$; let $a[t]$ stand on the place of the formal identifier $x[i]$ in the body of $p = ass(\pi'')$ in $\pi$. Then $d := \lambda(p) - \lambda(\varphi(x)) > 0$ and $a[t]$ is the $\nu$-th actual parameter of

$$stat(st^{d}(\pi'')) = \tilde{p}[t_o](\tilde{a}_1[t_1],...,\tilde{a}_n[t_n]), \quad \nu = X(x) .$$

Proof by induction:
Trivial for $\pi'' = \pi$.
Let $\pi' \vdash\!\!- \pi''$ and the lemma hold for $\pi'$ and its dynamic predecessors. x is not local to the body of p. Therefore

$$\lambda(\tilde{p}) \leq \lambda(p) \text{ and } 0 \leq d \leq \lambda(p) - \lambda(\tilde{p}) \leq \lambda(p)$$

1. case: x is the $\nu$-th formal parameter $x_\nu$ of p itself.
Then $\tilde{p} = p$, $d = 0$, $st^{d}(\pi'') = \pi''$, $a[t] = \tilde{a}_\nu[t_\nu]$ .

2. case: x is the $\nu$-th formal parameter $x_\nu$ of procedure $\tilde{p} = \varphi^d(p)$, $d > 0$. $a[t]$ occurs in $IGB(st(\pi''))$ with

$$st(\pi'') \overset{*}{\vdash\!\!\!-} \pi' \vdash\!\!\!- \tau''$$

By induction hypothesis $a[t]$ is the $\nu$-th actual parameter of

$$stat(st^{d-1}(st(\pi'')) = stat(st^{d}(\pi'')).$$

We now return to the question when a program $\pi$ does not fulfill the strong most recent property. Above we have seen that "$\pi$ does not fulfill the strong most recent property" is equivalent to

$(\exists \pi'' \in T_\pi \setminus \{\pi\}, \exists \mu$ with $0 \leq \mu \leq \lambda(ass(\pi'')))$

(a) $st^\mu(\pi'') \neq mr_\mu(\pi'')$,

b) $(\forall \pi' \in \vartheta(\pi''), \pi' \neq \pi'')(\nexists m$ with $0 \leq m \leq \lambda(ass(\pi')))$
$(st^m(\pi') = mr_m(\pi')))$

we have $\mu > 0$; otherwise $\mu = 0$ and

$st^\mu(\pi'') = \pi'' = mr_\mu(\pi'')$ (contradicts a) )

in addition: $st^\mu(\pi'') \overset{+}{\vdash\!\!\!-} mr_\mu(\pi'')$.

Let be $h := ass(st^\mu(\pi'')) = ass(mr_\mu(\pi''))$
$f := ass(\tau''); stat(\pi'') = f[t_o](...)$ in $\pi' \vdash\!\!\!- \tau''$
$g := ass(\pi'), \tau_o := \tau^{-1}(t_o)$

① h is formally recursive.

② f is declared in the body of h (because $h = \varphi^\mu(f), \mu > 0$)

③a $f[t_o]$ stands on the place of a formal identifier $x[\xi]$ in the main part of the body of g.

Proof: Assumption: $f[t_o]$ stands on the place of the non-formal identifier $f[\xi]$ in the main part of the body of g; then we have $\delta(\tau_o) \leq \delta(\pi')$ (Lemma 1) and therefore $\tau_o \in \delta(\pi')$ and $(\exists m)(st^\mu(\pi'') = st^m(\pi'))$.

Because of a) and $\mu>0$ we have
$(\exists\tilde{\pi})(h=ass(\tilde{\pi}),\ st^{\mu}(\pi'') \overset{+}{\vdash} \tilde{\pi} \overset{+}{\dashv} \pi' \vdash \pi'')$ and therefore
$st^{m}(\pi') \neq mr_{m}(\pi')$. This contradicts b).

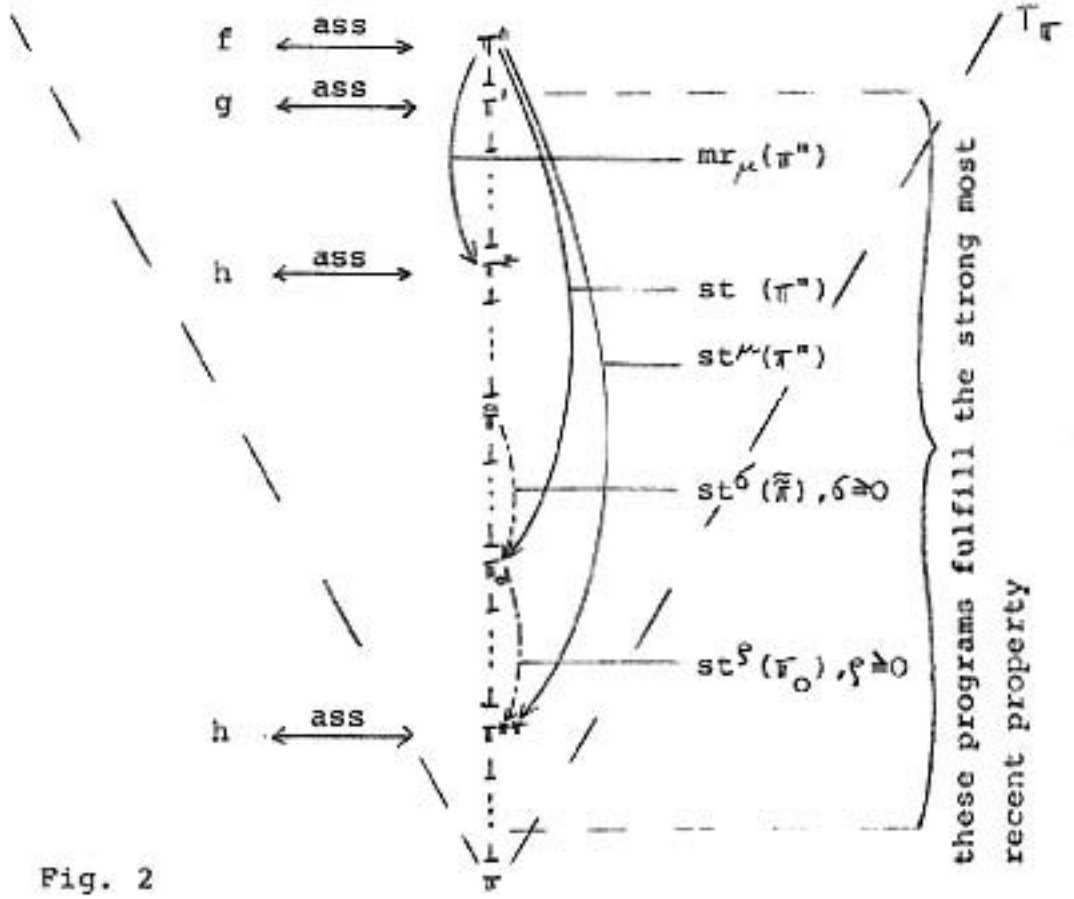



Fig. 2

(3b) The body of h contains a procedure statement q(...,f,...).

Proof: $f[t_o]$ is actual parameter of a procedure statement in $\tilde{\pi} \overset{+}{\dashv} \pi'$ and stands on the place of the non-formal identifier $f[\xi]$ in the main part of the body of $ass(\tilde{\pi})$.
$\sigma(\pi_o) \leq \sigma(\tilde{\pi})$ (Lemma 1); because of $\pi_o = st(\pi'')$ we have $st^{\mu}(\pi'') \in \sigma(\pi_o)$ and therefore $st^{\mu}(\pi'') \in \sigma(\tilde{\pi})$ (*),
i.e. $h = ass(st^{\mu}(\pi''))$ contains or is equal to $ass(\tilde{\pi})$.

(4) If q is non-formal then q formally calls h.

(5) If q is formal then a non-formal procedure $\bar{q}$ is formally inserted in q and $\bar{q}$ formally calls h.

Proof: Look at (*) from the proof of (3b):
$(\exists m \geq 0)(\ st^{\mu}(\pi'') = mr_m(\bar{\pi})$; because of b) no program $\hat{\pi}$ with $st^{\mu}(\pi'') \overset{+}{\vdash} \hat{\pi} \overset{\neq}{\vdash} \tilde{\pi}$ fulfills $ass(\hat{\pi}) = h$; so with (*) we have $st^{\mu}(\pi'') \overset{\neq}{\vdash} \tilde{\pi} \overset{+}{\vdash} mr_{\mu}(\pi'')$.

Collecting these results and bearing in mind Theorem 1 we get the following theorem which goes back to McGowan's theorem:

Theorem 2: If $\pi$ does not fulfill the strong most recent property then

1) there exists a potentially recursive procedure h;
2) the body of h contains the declaration of a procedure f;
3) the body of h contains a procedure statement q(...,f,...) having f as one of its actual parameters;
4) if q is non-formal then q potentially calls h;
5) if q is formal then a non-formal procedure $\bar{q}$ is potentially inserted in q and $\bar{q}$ potentially calls h.

The negation of 1),...,5) is a sufficient compile time decidable condition for the fact that $\pi$ fulfills the strong most recent property and, a fortiori, for the fact that $\pi$ fulfills the weak most recent property.

Examples: The programs $\pi_1$:

```
begin real a; proc h(x);{real b; proc f(y);{outreal b};
                         a:=a+1; b:=a; x(f)}; a:=1;h(h)    end
```

and $\pi_2$:

```
begin proc f(x);{proc g(y);{y(x)};
                 proc h(z);{z(h)}; h(g)};
      f(f)
end
```

have not the strong most recent property. $\pi_1$ fulfills 1),2),3),5) while $\pi_2$ fulfills 1),2),3),4).

## Concluding remarks

Mac Gowan's Theorem 2 could be formulated sharper; but we restrained because in [3] it has been proved that the strong and the weak most recent property both are compile time decidable.

## References

[1] Dijkstra, E., "recursive programming", in Programming systems and languages, S. Rosen, McGraw-Hill, New York,1967

[2] Grau, A.A., Hill, U., Langmaack,H., Translation of ALGOL 60, Handbook für Automatic Computation, Vol. Ib, Berlin-Heidelberg-New York, 1967

[3] Kandzia, P., "On the most recent property of ALGOL-like programs", in Proceedings of the Second Colloquium on Automata, Languages and Programming, Saarbrücken, July 29-August 2, 1974, Lecture Notes in Computer Science, Springer Berlin,Heidelberg,New York

[4] Langmaack, H., "On correct procedure parameter transmission in higher programming languages", Acta Informatica 2, 110-142, 1973

[5] McGowan, C.L., "the "most recent" error : its causes and correction", in Proceedings of an ACM conference on proving assertions about programs, SIGPLAN Notices, Vol. 7, Number 1, 1972

[6] Wegner, P., "Three computer cultures - computer technology, computer mathematics and computer science", Advances in Computers 10, 7-78, 1970

[7] Wegner, P., Programming languages, information structures, and machine organization, McGraw-Hill, New York, St. Louis, San Francisco, Toronto, London, Sydney, 1968