University of Amsterdam
Programming Research Group

Towards a Formalization of Budgets

J.A. Bergstra
S. Nolst Trenite
M.B. van der Zwaag

Report PRG0712 December 2007

J.A. Bergstra

Programming Research Group
Faculty of Science
University of Amsterdam

Kruislaan 403
1098 SJ Amsterdam
The Netherlands

tel. +31 20 525.7591
e-mail: janb@science.uva.nl

S. Nolst Trenite

Faculty of Science
University of Amsterdam

e-mail: sanne@science.uva.nl

M.B. van der Zwaag

Programming Research Group
Faculty of Science
University of Amsterdam

Kruislaan 403
1098 SJ Amsterdam
The Netherlands

tel. +31 20 525.7584
e-mail: mbz@science.uva.nl

Programming Research Group Electronic Report Series

Towards a Formalization of Budgets

Jan A. Bergstra[1] Sanne Nolst Trenité[2]
Mark B. van der Zwaag[1]

[1]Section Software Engineering, Informatics Institute, University of Amsterdam
[2]Faculty of Science, University of Amsterdam
Email: {janb,sanne,mbz}@science.uva.nl

Abstract

We go into the need for, and the requirements on, a formal theory of budgets. We present a simple algebraic theory of rational budgets, i.e., budgets in which amounts of money are specified by functions on the rational numbers. This theory is based on the tuplix calculus. We go into the importance of using totalized models for the rational numbers. We present a case study on the educational budget of a university department offering master programs.

1 Introduction

The process of budget design and financial accounting is becoming increasingly specialized and exclusive. Unfortunately, the need for an underlying theory seems to be unrecognized. Economic theories of finance do not provide the simple insights needed for managing small-scale operations. We are currently witnessing the following developments.

1. Financial work takes place in a context of complex IT support systems, which are often poorly documented from a user perspective. Documentation is typically limited to user manuals, and does not give conceptual descriptions of the underlying budget theory and financial theories.

2. Financial competence is easily confused with the ability to operate certain financial systems. Because these systems are increasingly complex, the competence to use them is becoming scarce and requires training and experience. Nevertheless that competence need not imply any deeper awareness of the variation of business logics that may or may not be served with a given system.

3. Financial planning is at the basis of many complex organizational transformations. Its logic is intimately connected with novel structural changes

such as outsourcing, insourcing, backsourcing and offshoring. Organizational changes are often correlated with changes in budget logic.

In this situation, we find it worthwhile to explore the applicability of modeling techniques developed in the fields of information science and software engineering. Unlike software architecture, financial architecture seems to be a subject to which relatively little attention is paid. It is worth an effort to apply system description techniques from computer science to financial systems and to facilitate systematic and correct reasoning about them. We believe that financial architecture can profit from the same development strategy as software architecture by making use of a basis of design patterns and by developing very clear modularization techniques.

In the other direction, we imagine a formalization of budgets to be a helpful ingredient for the development of sourcing theory (see for instance [5] and [6]). Sourcing theory requires the presence of so-called business cases for insourcer and outsourcer to be available and scrutinized before any deal is made. After outsourcing has been executed, services are expected to be delivered in accordance with an SLA (Service Level Agreement). Budget information is an essential part of any SLA. No complete theory of sourcing is possible without some theory of SLAs and their underlying budgets. SLAs constitute a relatively new topic in computing and their meta-theory is still in an initial stage. We expect that SLAs, or similar entities existing within a further evolved terminology, will become a central cornerstone of the emerging theory of service-oriented computing.

Overview. In this article we define a simple algebraic theory of *rational budgets*, that is, budgets in which amounts of money are specified by functions on the rational numbers. For this theory we borrow from our experience with process algebras (modeling the behavior of computer systems) and abstract data types. In Section 2 we go into the use of rational functions, and in particular on the importance of *totalized* models for the rational numbers when defining them as an abstract data type. In Section 3 we reflect on the formalization of budgets, and this is followed by a simple theory of budgets in Section 4. We use the so-called tuplix calculus [2] to model budgets. In Section 5 we present a case study on the educational budget of a university department offering master programs.

2 Rational Numbers

It is a common misunderstanding that because budget figures are to be understood as measuring quantities of money expressed in terms of known currencies all semantic problems will disappear. The issue is comparable to a program notation designed for programming computations on natural numbers. In spite of the seemingly clear mathematical basis of the program notation, in the absence of a formally specified semantics of the program notation at hand, nothing much can be said about what transformations on natural numbers a specific program denotes.

For the definition of a budget the data type of rational numbers is considered of central importance. All financial quantities will me measured in exact rational numbers. If the question arises what exactly are the rational numbers we refer to [3] which provides a novel and concise initial algebra specification of this classical mathematical structure. We hold that division plays an important role in budgeting, because of the need to distribute expected costs over a number of expected users. If ten users will make mutually comparable use of a single shared service each of them will be expected to pay 10% of its costs unless more specific information is available. Interestingly, division seems not to feature in accounting and bookkeeping. In [3], division is given a first class status with an operator symbol reserved for it, very much like addition, multiplication and subtraction. Moreover we will insist, against conventional practice, that division is a total operator. This leads to 'meadows' of rational numbers: a meadow is the well-know algebraic structure 'field' with a total operator for division, so that division by zero produces some value in the domain of the field. In a zero-totalized field division is made total by choosing zero as the result of division by zero (and, for example, in a 47-totalized field one has chosen 47 to represent the result of all divisions by zero).

The relevance for our theory of budgets is this: budgets will contain expressions for rational functions rather than 'closed' figures in full (fixed point) precision. From the conventional perspective on rational numbers, these functions may be undefined for certain input values, namely, for values leading to devision by zero. In general, it will be far from trivial to decide whether functions are always defined, and, if not, to establish the values for which it is. On the other side, the meadow of rational numbers constitutes a total algebra with trivial type-checking properties, that provides us with a clear meaning of expressions. This will be of the highest importance to our endeavor, because just like in the case of specifying computer programs, there will be no way to avoid explicit syntax and type-checking of expressions.

A down-side of working with zero-totalized fields is that some calculations will produce useless results. In most cases the occurrence of division by zero in the course of a calculation still indicates the presence of an error somewhere and error detection techniques will be needed. Nevertheless the meta-theory of this form of error detection is considered far simpler than the meta-theory of partial algebras, thus creating a trade-off to the advantage of the use of calculation in zero-totalized fields.

A small digression: one might wonder why the issue to define division by zero is so easily avoided in school mathematics and its academic sequel. The answer is that in mathematics most specialists make no distinction between syntax and semantics. No syntactic expression is entitled to any attention on the sole grounds of its formal existence. If syntax is used its use follows the development of semantics and there always is an intended meaning. Realistically, the question 'what is the intended meaning of that piece of syntax' cannot be even posed. A reluctance to separate syntax and semantics may become a weakness when concepts need to be defined which are viewed as constructs of a syntactic nature consisting of parts rooted in classical mathematics.

3 Theory and Practice of Budgets

A crucial point in the design of a formal theory of budgets is the following separation of concerns. Our task is to give a conceptual, mathematical definition of budgets. This definition (of what a budget *is*) should be as much as possible independent of how and why budgets are *used*. A budget will not be assigned a behavior of its own a priori. One wants to avoid definitions like 'natural numbers are a very practical concept that has been in use since the need arose to count sheep, for which natural numbers turn out to work very well fortunately.' Clearly budgets are artifacts of a human origin, and, as in the case of natural numbers or data bases, their use is independent of the artifact at hand.

We believe that a formal theory of budgets is essential to the analysis and improvement of their use. Some examples:

(1) The practice of budget design can be compared to the practice of computer programming: in general programmers have no means available to know in advance what computers will do with their writings. This uncertainty is mainly due to a lacking theoretical basis but equally to a common ethos which acts against the use of that theoretical basis even if it happens to exist and if it might be readily available at reasonable costs. In computer programming, testing is the main though unconvincing tool to fight this form of uncertainty. In budget design the concept of testing is significantly harder to imagine, however. Budgets seem to be submitted to a number of static checks only. Then after their use some form of evaluation and assessment may produce new guidelines (design rules) for budgeting and new budgets will be matched with these new design rules as well. Budget testing comparable to dynamic testing of control code will require a simulation environment. That environment is quite specific to an organization and most organizations have no such tool available to them at the time of this writing. Clearly, budget simulation is a hopeless endeavor without a formal theory of budgets.

(2) Irrespective of the objectives a budget designer has in mind when writing a budget, it will evolve through a life-cycle. An organization may prescribe this life-cycle to its budget designers in very much the same way as a software life-cycle may play a normative role in a computer software production factory. Like a machine control code can be active (running, executing) a budget can be used to control events within an organization. We would hope that when we start with a clear formal definition, we may be able to explain (at least in principle) how that form of control might work. Also, as budgets are often very context-specific, they may be compared to dedicated computer programs. Renewing a budget on an annual basis may be compared to computer software maintenance (although that comparison may well underestimate the degree of innovation that a new budget requires).

(3) When hard-pressed to qualify the writing of a budget, the following viewpoint might be reasonable: budgets are proposed in a context in which their proposal is best viewed as a move from the side of its author in a game which is implicitly present in the mentioned context. Let us assume that a budget is designed for an activity called A. After having been designed and

worked out in detail, the distribution of a budget can be considered a move in a game. By introducing a budget proposal the decision-making process is somehow influenced. One may assume that this process will eventually lead to a validated budget for activity A. The budget proposal may impact the style of budget design which will be adopted for A and for similar activities. The intriguing observation is that when time has come to write budgets for A, many different and competing budget proposals may be simultaneously put forward. Thus at budget design time there is no such thing as 'the budget' in very much the same way as a computer program under construction leaves open many degrees of freedom. It may be fruitful to experiment with writing quite different budgets for the financial control of a single activity A. Again, such analysis of the practice of budgeting should start with a formal theory of budgets.

4 A Simple Budget Algebra

We present a simple algebra for rational budgets. This algebra is an application of the so-called tuplix calculus [2]. A tuplix (plural: tuplices) is a datastructure that collects attribute-value pairs. The tuplix calculus provides signature and axioms for various operations on budgets, including a means to express constraints on budgets, the composition of budgets, and encapsulation. So, budgets will be given by means of tuplix expressions. The design of user-friendly syntax for budget expressions is outside the scope of this paper. However, the definition of budgets as tuplices provides a rudimentary syntax which will suffice for explanatory purposes.

4.1 Entries and Tests

The basic building blocks of budgets are *entries* and *tests*. An entry is an attribute-value pair of the form

$$a(p)$$

where a is an attribute from a given set A of attribute symbols, and p is a data term. For the values we use the data type of rational numbers, which we assume to be given by a zero-totalized field as explained in Section 2 (so p/q is always defined). An entry represents a payment: the attribute is used in the communication between payer and payee, and describes or identifies a transaction; we refer to the attribute as the *channel* of the transaction, and shall also say that the payment occurs *along* the channel. The term p represents the amount of money involved. An entry $a(p)$ with $p > 0$ stands for an obligation to pay amount p along channel a. If $p < 0$, the entry stands for the expected receipt of amount p along a.

A *zero test* is a term of the form

$$\gamma(p)$$

for amount p. It acts as a conditional: if the argument p equals zero, then the test is void and disappears from compositions; if the test is not equal to zero, it nullifies any composition containing it. Observe that an equality test $p = q$ can be expressed as $\gamma(p - q)$.

4.2 Budget Composition

We define a budget as a (conjunctive) composition of entries and zero tests. This composition is commutative and associative:

$$x \oplus y = y \oplus x, \tag{1}$$
$$(x \oplus y) \oplus z = x \oplus (y \oplus z). \tag{2}$$

There are two constants for budgets: the *empty budget*, notation ε, and the *null budget*, notation δ. The empty budget stands for the absence of entries or tests, and the null budget is used to model an erroneous situation which nullifies the entire composition containing it. Axioms:

$$x \oplus \varepsilon = x, \tag{3}$$
$$x \oplus \delta = \delta. \tag{4}$$

Entries with the same attribute can be combined:

$$a(u) \oplus a(v) = a(u + v). \tag{5}$$

Note in particular that the composition of a payment $a(p)$ and the receipt $a(-p)$ can be reduced to $a(0)$. We shall see below that *encapsulation* both enforces and hides such synchronizations.

For the axiomatization of the zero tests, we use the property that in the zero-totalized field for the rational numbers, the division p/p yields zero only if p is equal to zero; otherwise it yields 1. Axioms:

$$\gamma(u) = \gamma(u/u), \tag{6}$$
$$\gamma(0) = \varepsilon, \tag{7}$$
$$\gamma(1) = \delta. \tag{8}$$

For reasoning about budgets with open data terms, we add the following two axioms:

$$\gamma(u) \oplus \gamma(v) = \gamma(u/u + v/v), \tag{9}$$
$$\gamma(u - v) \oplus a(u) = \gamma(u - v) \oplus a(v). \tag{10}$$

4.3 Encapsulation

For set of attributes $H \subseteq A$, the operator $\partial_H(x)$ encapsulates all entries with attribute $a \in H$ occurring in x. That is, if the accumulation of quantities in entries with attribute a equals zero, the encapsulation on a is considered successful and the a-entries disappear; if the accumulation is not equal to zero, it yields the null budget δ. Axioms:

$$\partial_H(\varepsilon) = \varepsilon, \tag{11}$$

$$\partial_H(\delta) = \delta, \tag{12}$$

$$\partial_H(\gamma(u)) = \gamma(u), \tag{13}$$

$$\partial_H(a(u)) = \begin{cases} \gamma(u) & \text{if } a \in H, \\ a(u) & \text{if } a \notin H, \end{cases} \tag{14}$$

$$\partial_H(x \oplus \partial_H(y)) = \partial_H(x) \oplus \partial_H(y). \tag{15}$$

We further adopt the identities

$$\partial_{H \cup H'}(x) = \partial_H \circ \partial_{H'}(x) \quad \text{and} \quad \partial_\emptyset(x) = x.$$

Example. Consider budget

$$P \stackrel{\text{def}}{=} a(-30) \oplus b(10) \oplus b(20).$$

This budget specifies the expected receipt of amount 30 along channel a and payments of amount 10 and of amount 20 along b. We compose it with budget

$$Q \stackrel{\text{def}}{=} b(-30) \oplus c(30),$$

which specifies that amount 30 is received along b and sent along channel c. We see that the payments of P will match the receipt of Q on channel b, so that encapsulation of b will hide these entries:

$$\partial_{\{b\}}(P \oplus Q) = a(-30) \oplus c(30).$$

To derive this, first derive that

$$\partial_{\{b\}}(a(-30) \oplus c(30)) = a(-30) \oplus c(30)$$

using axioms (14) and (15). Then:

$$\begin{aligned}
\partial_{\{b\}}(P \oplus Q) &= \partial_{\{b\}}(a(-30) \oplus b(10) \oplus b(20) \oplus b(-30) \oplus c(30)) \\
&= \partial_{\{b\}}(b(0) \oplus a(-30) \oplus c(30)) \\
&= \partial_{\{b\}}(b(0) \oplus \partial_{\{b\}}(a(-30) \oplus c(30))) \\
&= \partial_{\{b\}}(b(0)) \oplus \partial_{\{b\}}(a(-30) \oplus c(30)) \\
&= \gamma(0) \oplus a(-30) \oplus c(30) \\
&= a(-30) \oplus c(30).
\end{aligned}$$

Another example. When computing the encapsulation of more than one channel, we split the encapsulation up, and compute them one by one. Recall that we defined

$$\partial_{H \cup H'}(x) = \partial_H \circ \partial_{H'}(x).$$

For example, we derive

$$\partial_{\{a,b\}}(a(0) \oplus b(0)) = \partial_{\{a\}} \circ \partial_{\{b\}}(a(0) \oplus b(0)) = \partial_{\{a\}}(a(0)) = \varepsilon.$$

4.4 Constraints

In the case study in this article, we assume that an absolute operator $|\_|$ (defined by $|p| = p$ if $p \geq 0$, and $|p| = -p$ otherwise) is part of the signature for rationals. With this operator we can express inequalities:

$$\gamma(|q - p| - (q - p))$$

expresses the test $p \leq q$. For inequality tests we shall then simply write $\gamma(p \leq q)$. We sometimes write $\gamma(p = q)$ for $\gamma(p - q)$.

For example, we may design a budget under the constraint

$$\phi \stackrel{\text{def}}{=} p \leq q,$$

and we then compose the budget with the test $\gamma(\phi)$. For the composition under multiple constraints, say ϕ and ψ, we may use the notation

$$\gamma(\phi \wedge \psi) \stackrel{\text{def}}{=} \gamma(\phi/\phi + \psi/\psi) \stackrel{(9)}{=} \gamma(\phi) \oplus \gamma(\psi).$$

5 Case Study: MSc Program Budgets

We consider a university department that maintains the three MSc programs A, B and C. Each program offers a 1-year, 60 EC[1] curriculum. These programs need a new budget because of changes concerning budget guidelines, financial reporting, risk management and business accounting. Below we develop budgets for the programs. Having fixed these budgets, the three program managers should negotiate the setting of certain variables. Having done that the program managers are free to develop their programs within the constraints of the budget. This process may be viewed as a game aimed at the design of a single budget where coalitions try to get things their way by imposing preferred variable settings on other participants. In our case, the program managers will not hesitate to get money their way at the expense of the other programs or to prove other programs financially unsound, should they find possibilities to do so in the new system.[2]

5.1 Generic Structure of an MSc Program

Each of the three programs has the following structure:

1. An introduction week providing general information.

[1] An EC is a unit of student activity/learning outcome in the European Credit Transfer System. One EC stands for 28 hours of work.

[2] Mark Burgess (see for instance [4]) advocates the mechanism of autonomous agents making promises which constrain their actions on a voluntary basis only. A budget proposal might be viewed as a promise conditional under the counter promise by other parties that they will go along with it. Finding protocols for distributed budget design in a context of voluntary cooperation is an interesting challenge for further research.

2. 4 courses of 10 EC each. A course consists of 300 working hours composed from these ingredients:
   - Between 40 and 160 hours of teaching by senior staff.
   - Working group meetings supervised by junior staff.
   - Unsupervised student team meetings.
   - Unsupervised individual experimental work.
   - Unsupervised individual homework.
   - Participation in an examination.
3. 2 projects of 10 EC each. A project is supervised in one of the following ways:
   - Between 5 and 10 hours of internal senior staff supervision.
   - At most 20 hours of junior staff supervision.
   - At most 5 hours of external staff supervision (performed outside the institution).

   A project ends with a 30 minute presentation (at least two senior staff members present).
4. A formal final degree ceremony.

Each program offers at least two mandatory courses for its own students, and may offer a number of optional courses that can also be followed by students from the other programs. An expensive way to implement this is to offer six dedicated courses in the program and to make two of these compulsory while leaving the students the option to choose two from the other four courses. A reason to use this kind of planning may be to let research staff lecture about their advanced topics in order to recruit future PhD students. Another reason might be to make sure that the entire student population acquires a wide body of knowledge representative of the field as a whole while accepting that each individual student has acquired knowledge in a more limited scope. A much cheaper option is to offer only the 2 mandatory courses and to ask the students to take electives from courses offered by other programs. Reason for doing so may be a lack of staff or financial resources. Another reason might be the intention to educate a homogeneous group of experts who will be able to cooperate effectively in forthcoming projects.

5.2 Joint Budget

In this case study we specify four budgets: budgets A, B, and C, for the respective programs A, B, and C, and one joint budget J. We start with the joint budget.

All income of the programs from external sources is specified in the joint budget, and these incoming amounts are received via channel *in*. The task of the joint budget is to specify the distribution of the income between the three programs and shared costs. Payments from the joint budget to the individual program budgets run via the respective channels a, b, and c. The only shared costs are the payments to the so-called educational service center (student consulting, time tabling, lecture hall reservation, facility management, administration); these payments are done via channel e. Picture:

$$\begin{tikzcd}
 & {} \arrow[d, "in"'] & {} \\
 & J \arrow[dl, "a"'] \arrow[d, "b"] \arrow[dr, "c"] \arrow[ur, "e"] & \\
A & B & C
\end{tikzcd}$$

When we consider the composition of the four budgets, payments along the channels a, b and c are considered to be internal. The channels in and e are external; they 'link' to parties for which we do not have the budgets.

Notation. The letter X ranges over A, B, C (denoting the programs). We use lowercase italics for variable names, and uppercase roman for abbreviations.

Variables. The variables used in the specification of the joint budget J are listed in Figure 1. The values for the variables $X{:}nec$ and $X{:}ndg$ are determined by measurement and monitoring (in practice one may take last years numbers instead). The values for the variables $bbpp$ and k are determined by negotiation between the program managers. We shall return to the consequences of the setting of these variables.

Income. The income, received via channel in, consists of two parts.

First, there is the overall EC compensation (ECC), defined as the total number of EC credits awarded in the three programs, multiplied by the compensation per credit:

$$\mathrm{NEC} \stackrel{\mathrm{def}}{=} \sum_X X{:}nec,$$

$$\mathrm{ECC} \stackrel{\mathrm{def}}{=} \mathrm{NEC} \cdot cpec.$$

Similarly, there is the overall degree compensation (DGC), defined as the total number of degrees awarded in the three programs, multiplied by the compensation per degree:

$$\mathrm{NDG} \stackrel{\mathrm{def}}{=} \sum_X X{:}ndg,$$

$$\mathrm{DGC} \stackrel{\mathrm{def}}{=} \mathrm{NDG} \cdot cpdg.$$

Expenses. The educational service center (ESC) takes care of all data base handling, time tabling, logistics, communication and marketing, help desks of various kinds, international relations and formal ceremony management. There

Set by external authority:

cpec	The *compensation per EC*. Amount obtained when awarding one EC.
cpdg	The *compensation per degree*. Amount obtained when awarding one degree.
escf	The *educational service center fraction* (value between 0 and 1) of the overall income to be transferred to the educational service center.

Set by measurement:

X:*nec*	Total number of EC awarded in courses offered by program X.
X:*ndg*	Number of degrees awarded in program X.

Set by budget designer/negotiation:

bbpp	A fixed amount that serves as the *basic budget per program*. This amount is equal for each program and is used to pay for the program manager, various committee tasks, marketing and communication.
k	Fraction (value between 0 and 1) of the *bbpp* which is taken from the part of the overall income stemming from degree compensation. The remaining fraction $1 - k$ of the *bbpp* is taken from the overall EC compensation.

Figure 1: Variables used in budget J

is a joint payment to the ESC, consisting of the fraction *escf* of the overall income:

$$\mathrm{ESC} \stackrel{\text{def}}{=} \mathit{escf} \cdot (\mathrm{ECC} + \mathrm{DGC}).$$

The remainder $(1 - \mathit{escf}) \cdot (\mathrm{ECC} + \mathrm{DGC})$ of the overall income is distributed among the three programs.

First, each program receives the amount *bbpp*. Of course, this amount cannot be more than one third of the available money, so we adopt the constraint

$$\phi_1 \stackrel{\text{def}}{=} \mathit{bbpp} \leq (1/3) \cdot (1 - \mathit{escf}) \cdot (\mathrm{ECC} + \mathrm{DGC}).$$

Furthermore, we require that fraction k of the *bbpp* is taken from the overall degree compensation, and the remaining part $(1 - k)$ of the *bbpp* is taken from the overall EC compensation, leading to the following constraints:

$$\begin{aligned}
\phi_2 &\stackrel{\text{def}}{=} k \leq (\mathrm{DGC} \cdot (1 - \mathit{escf}))/(3 \cdot \mathit{bbpp}), \\
\phi_3 &\stackrel{\text{def}}{=} (1 - k) \leq (\mathrm{ECC} \cdot (1 - \mathit{escf}))/(3 \cdot \mathit{bbpp}).
\end{aligned}$$

Apart from the fixed amount *bbpp*, that provides each program with a financial basis independent of its own student numbers (assuming that the other programs have sufficient numbers of students), each program gets a share of the remaining part of the overall EC and degree compensation. These shares are proportional to the contribution that the program has in the overall compensation. The remaining part of the degree compensation, after subtraction of the expenses on ESC and *bbpp*, is

$$\mathrm{DGC} \cdot (1 - \mathit{escf}) - 3 \cdot k \cdot \mathit{bbpp},$$

and the share of this amount awarded to program X is $X{:}\mathit{ndg}/\mathrm{NDG}$. So we define

$$X{:}\mathrm{DGC} \stackrel{\text{def}}{=} (\mathrm{DGC} \cdot (1 - \mathit{escf}) - 3 \cdot k \cdot \mathit{bbpp}) \cdot X{:}\mathit{ndg}/\mathrm{NDG}.$$

Similarly, we define

$$X{:}\mathrm{ECC} \stackrel{\text{def}}{=} (\mathrm{ECC} \cdot (1 - \mathit{escf}) - 3 \cdot (1 - k) \cdot \mathit{bbpp}) \cdot X{:}\mathit{nec}/\mathrm{NEC}.$$

Each program X receives from the joint budget the amount

$$X{:}\mathrm{STAFF} \stackrel{\text{def}}{=} \mathit{bbpp} + X{:}\mathrm{DGC} + X{:}\mathrm{ECC}.$$

Budget. Putting everything together, the joint budget J is defined by

$$\begin{aligned}
J \stackrel{\text{def}}{=} \gamma(\phi) \oplus \mathit{in}(-\mathrm{ECC}) \oplus \mathit{in}(-\mathrm{DGC}) \oplus e(\mathrm{ESC}) \oplus \\
a(A{:}\mathrm{STAFF}) \oplus b(B{:}\mathrm{STAFF}) \oplus c(C{:}\mathrm{STAFF}),
\end{aligned}$$

where

$$\phi \stackrel{\text{def}}{=} \phi_1 \wedge \phi_2 \wedge \phi_3.$$

Notes. The joint budget has been designed with the following properties in mind.

- By taking $bbpp$ low (or simply zero) each budget gets as much as possible resources proportional to its production in EC and in degrees. By taking $bbpp$ higher each program budget is provided with a minimum funding with the effect that each program gets less return on investment for a single EC or degree.

  For example, a program with relatively few students may strive for a significant fixed budget basis $bbpp$ (maybe even $bbpp = (1/3) \cdot (1 - escf) \cdot (\text{DGC} + \text{ECC})$ in an extreme case).

- By taking k low (or simply zero) a maximal reward is provided for programs with a high yield in terms of degrees. By taking k high (or simply 1) a maximal reward is given to programs that get as many as possible ECs to students irrespective of their program and irrespective of whether or not they will complete their degree.

  For example, again for a program with relatively few students: choose fraction k as close as possible to 1 (for all values of $bbpp$,

  $$k = (1/3) \cdot (\text{DGC} \cdot (1 - escf))/bbpp$$

  seems to be a reasonable choice). By taking k as close as possible to 1, the program will profit the most from students from the other programs following its courses.

- The following constraints need not be imposed as they follow from the defining equations, that is by adding these constraints the meaning of the budget will not change:

  $$\sum_X X{:}\text{DGC} = \text{DGC} \cdot (1 - escf) - 3 \cdot k \cdot bbpp,$$
  $$\sum_X X{:}\text{ECC} = \text{ECC} \cdot (1 - escf) - 3 \cdot (1 - k) \cdot bbpp.$$

5.3 Budgets per Program

In the individual program budgets, the amount received from the joint budget is spent on the following costs:

- Senior Educational Staff (SES). Compensation for educational working hours by senior staff.

- Junior Educational Staff (JES). Compensation for educational working hours by junior staff.

- Program Management (PM). Compensation for all forms of program management performed by educational staff.

Set by external authority:	
sscph	Senior staff marginal integral cost per hour.
jscph	Junior staff marginal integral cost per hour.
Set by program manager:	
X:*lpf*	Lecture preparation factor (the number of hours used to prepare one hour of lecturing).
X:*sset*	Senior staff examination time: time needed to set and mark an exam.
X:*sspst*	Senior staff project supervision time (number of hours spent by senior staff supervising a single student project).
X:*jspst*	Junior staff project supervision time (number of hours spent by junior staff supervising a single student project) in addition to senior staff supervision.
X:*ssot*	Senior staff overhead in total (hours per year).
X:*pmt*	Program management time (hours per year spent by program manager).
X:C:*sslt*	Senior staff lecturing time (number of hours) for course C.
X:C:*jsst*	Junior staff supervision time (number of hours) for course C.

Figure 2: Variables used in program budgets

Variables. The variables used in the program budgets are listed in Figure 2. Within the constraints set by the educational budget, a program manager can vary the values for the second set of variables. Needless to say this leads to a combinatorial explosion of options. Setting these variables low implies a sound budget but introduces risk with student success rates, with student satisfaction monitoring and periodically with external quality control authorities. Most importantly, however, setting the other variables very low will cause senior staff to complain about unrealistic requirements and workloads.

Budgets. Define the senior staff working hours (SSH) and junior staff working hours (JSH) as follows, with C ranging over the set C_X of courses offered by program X.

$$X{:}\mathrm{SSH} \stackrel{\mathrm{def}}{=} \sum_C (X{:}C{:}\mathit{sslt} \cdot (1 + X{:}\mathit{lpf}) + X{:}\mathit{sset}) + X{:}\mathit{ndg} \cdot 2 \cdot X{:}\mathit{sspst}$$

$$X{:}\mathrm{JSH} \stackrel{\mathrm{def}}{=} \sum_C (X{:}C{:}\mathit{jsst}) + X{:}\mathit{ndg} \cdot 2 \cdot X{:}\mathit{jspst}$$

The factor 2 in the summands for project supervision stems from the fact that each student does two projects. Note that project supervision generates compensation only when students obtain their degree. Failed projects or projects for students failing elsewhere in the program will not generate financial resources.

The expenses for senior and junior staff are found by multiplying their hours by their respective marginal integral costs per hour:

$$X\text{:SES} \stackrel{\text{def}}{=} X\text{:SSH} \cdot sscph,$$
$$X\text{:JES} \stackrel{\text{def}}{=} X\text{:JSH} \cdot jscph.$$

Finally, the staff costs for program management are given by

$$X\text{:PM} \stackrel{\text{def}}{=} (X{:}ssot + X{:}pmt) \cdot sscph.$$

Budgets.

$$A \stackrel{\text{def}}{=} a(-A\text{:SES} - A\text{:JES} - A\text{:PM})$$
$$B \stackrel{\text{def}}{=} b(-B\text{:SES} - B\text{:JES} - B\text{:PM})$$
$$C \stackrel{\text{def}}{=} c(-C\text{:SES} - C\text{:JES} - C\text{:PM})$$

Notes.

1. Both senior staff members and junior staff members may spend two kinds of hours: regular office hours and spare time hours. The second kind of work is unpaid. No one will be ever forced to work without compensation but a culture may exist where this is done on a regular basis. It is quite common to perform unpaid research work outside regular hours. This is possible for teaching just as well. Nevertheless there are some constraints. All formal teaching, all introductory activity, all examinations and all senior and junior staff supervision must take place within office hours and the cost of this staff time is given by fixed rates per hour. Office hours are either classified as educational office hours or as research office hours or as unclassified office hours. If educational activity is performed in research office hours it need not be paid from the educational budget but it will be paid from the research budget instead. This mechanism allows a research group to subsidize its teaching activities from its research budget. That subsidy may be justified in the case educating a small group of students brings about a few PhD students who might be very hard to find otherwise.

2. In these budgets per program a formidable amount of freedom exists because all variables determining the amount of junior and senior staff time spent on courses and projects can me defined specifically for each of the programs. If projects are very close to staff research they may be supervised in (seemingly) less time because additional unspecified research time is used for the supervision as well while those hours are not paid from this educational budget. If for instance program A has rather few students in comparison with the other two the following variable settings (or rather suggestions for setting variables) can be helpful:

(a) Set low senior staff hours for project supervision, and compensate that setting with higher junior staff hours and with time paid for from the research budget which is not viewed as a part of these budgets.

(b) Reduce the number of contact hours in course lecturing while requiring a significant amount of autonomous work from the students and setting difficult exam papers. This strategy may backfire when students from the other programs are supposed to attend the same lectures, however.

5.4 Synchronization of the Budgets

We have presented the four budgets A, B, C, and J. The combined budget is the synchronization

$$\partial_{\{a,b,c\}}(A \oplus B \oplus C \oplus J).$$

We derive that it equals

$$\gamma(\phi \wedge \psi_A \wedge \psi_B \wedge \psi_C) \oplus in(-(\mathrm{DGC} + \mathrm{ECC})) \oplus e(\mathrm{ESC})$$

where

$$\psi_X \stackrel{\text{def}}{=} (X{:}\mathrm{STAFF} = X{:}\mathrm{SES} + X{:}\mathrm{JES} + X{:}\mathrm{PM}).$$

Conclusion. This example shows how a modular decomposition of a budget can be designed. The decomposition is valid under a number of conditions only. If these conditions are not met, further refinement of the budgets is needed. That can be done by means of the same notation of course.

These conditions describe an abstraction level in the sense that they rule out circumstances which may be of practical interest but which are considered an undesirable overhead at a certain stage of design.

5.5 Further Reflections

Methods of Cost Measurement. Even if one observes the course of action when a particular program is run in all detail it is still difficult to make a precise statement concerning its costs. A first difficulty is how to count or incorporate free time hours made by staff members. A second difficulty is how to decide if any official research time is used for educational purposes. A third issue is to determine the border between research and preparation for lectures and project supervision. A further complication for cost measurement based on observations on how the work is actually done is this: if staff members are made aware of how their time investment is counted they may change their behavior. Suppose a staff member often works additional unpaid hours at home to get research done and then finds out that a counting system has detected many hours spent on teaching within the institution. This may lead to the conclusion that the position will be reclassified into a teaching position with a limited research task only. Then of course this staff member may be inclined to interchange a number of educational support activities (marking exams, preparing lectures, etc) with research activities that were done outside the office hours. Thus a counting system should be stable in the sense that its introduction should not by itself influence staff behavior in such a way that the results of counting are modified. In order to obtain this form of stability staff members should be given a variety of options for formally accounting their time.

Budgets versus Costs and Planning. Given the combinatorial explosion of options for planning an MSc program in the formats given above it is an unreasonable request to 'offer the program in a cheaper form' unless very clear goals are stated in advance. The way in which a curriculum (that is, the listing of course titles and of project proposals) can be offered and planned implies that it is useless to suggest a change on financial grounds unless a quite clear model of costs and revenues is available and unless a clear target in that model has been set in advance.

6 Conclusion

We have motivated our interest in a formal theory of budgets, and we have proposed a simple algebraic theory of budgets based on the tuplix calculus [2]. Quantities are expressed as functions on the rational numbers, which we have modeled as a totalized field [3]. As a case study, we modeled budgets and their composition for a university department offering master programs. We have kept the theory simple, but think of extensions such as operators for choice (as present in the tuplix calculus), binding of rational variables, a theory of interfaces and hiding, etc.

As a preliminary conclusion we state that budgets are amenable to formalization in the the data type tradition of theoretical computer science. The example demonstrates that formalization can be helpful to specify details which are likely to be missed in a less formal treatment and which are helpful for a proper understanding. At the same time, while working on the example, we have drawn the conclusion that designing budgets in a modular fashion is not an obvious matter and that many more cases studies will be needed to obtain a stable and formalized structure theory of budgets.

References

[1] J.A. Bergstra and A. Ponse. Interface groups for financial transfer architectures. `www.arxiv.org`, arXiv:0707.1639v1, July 2007.

[2] J.A. Bergstra, A. Ponse, and M.B. van der Zwaag. Tuplix Calculus. Electronic report PRG0713, Section Software Engineering, University of Amsterdam, December 2007. Available at `http://www.science.uva.nl/research/prog/`.

[3] J.A. Bergstra and J.V. Tucker. The rational numbers as an abstract data type. *Journal of the ACM* 54(2), 2007.

[4] M. Burgess. An approach to understanding policy based on autonomy and voluntary cooperation. *16th IFIP/IEEE Distributed Systems Operations and Management (DSOM 2005)*, LNCS 3775, Springer-Verlag, 2005.

[5] G.P.A.J. Delen. *Decision- en controlfactoren voor sourcing van IT* (in Dutch). PhD Thesis, University of Amsterdam (van Haren Publishing Zaltbommel), 2005.

[6] D.B.B. Rijsenbrij and G.P.A.J. Delen. Enterprise-architectuur is een noodzakelijke voorwaarde voor verantwoorde outsourcing (in Dutch). In *IT service management best practices* (red. J. van Bon), van Haren Publishing Zaltbommel, 35–58, 2004.

Electronic Reports Series of the Programming Research Group

Within this series the following reports appeared.

[PRG0711] J.A. Bergstra and C.A. Middelburg, *Program Algebra with a Jump-Shift Instruction,* Programming Research Group - University of Amsterdam, 2007.

[PRG0710] J.A. Bergstra and C.A. Middelburg, *Instruction Sequences with Dynamically Instantiated Instructions,* Programming Research Group - University of Amsterdam, 2007.

[PRG0709] J.A. Bergstra and C.A. Middelburg, *Instruction Sequences with Indirect Jumps,* Programming Research Group - University of Amsterdam, 2007.

[PRG0708] B. Diertens, *Software (Re-)Engineering with PSF III: an IDE for PSF,* Programming Research Group - University of Amsterdam, 2007.

[PRG0707] J.A. Bergstra and C.A. Middelburg, *An Interface Group for Process Components,* Programming Research Group - University of Amsterdam, 2007.

[PRG0706] J.A. Bergstra, Y. Hirschfeld, and J.V. Tucker, *Skew Meadows,* Programming Research Group - University of Amsterdam, 2007.

[PRG0705] J.A. Bergstra, Y. Hirschfeld, and J.V. Tucker, *Meadows,* Programming Research Group - University of Amsterdam, 2007.

[PRG0704] J.A. Bergstra and C.A. Middelburg, *Machine Structure Oriented Control Code Logic (Extended Version),* Programming Research Group - University of Amsterdam, 2007.

[PRG0703] J.A. Bergstra and C.A. Middelburg, *On the Operating Unit Size of Load/Store Architectures,* Programming Research Group - University of Amsterdam, 2007.

[PRG0702] J.A. Bergstra and A. Ponse, *Interface Groups and Financial Transfer Architectures,* Programming Research Group - University of Amsterdam, 2007.

[PRG0701] J.A. Bergstra, I. Bethke, and M. Burgess, *A Process Algebra Based Framework for Promise Theory,* Programming Research Group - University of Amsterdam, 2007.

[PRG0610] J.A. Bergstra and C.A. Middelburg, *Parallel Processes with Implicit Computational Capital,* Programming Research Group - University of Amsterdam, 2006.

[PRG0609] B. Diertens, *Software (Re-)Engineering with PSF II: from architecture to implementation,* Programming Research Group - University of Amsterdam, 2006.

[PRG0608] A. Ponse and M.B. van der Zwaag, *Risk Assessment for One-Counter Threads,* Programming Research Group - University of Amsterdam, 2006.

[PRG0607] J.A. Bergstra and C.A. Middelburg, *Synchronous Cooperation for Explicit Multi-Threading,* Programming Research Group - University of Amsterdam, 2006.

[PRG0606] J.A. Bergstra and M. Burgess, *Local and Global Trust Based on the Concept of Promises,* Programming Research Group - University of Amsterdam, 2006.

[PRG0605] J.A. Bergstra and J.V. Tucker, *Division Safe Calculation in Totalised Fields,* Programming Research Group - University of Amsterdam, 2006.

[PRG0604] J.A. Bergstra and A. Ponse, *Projection Semantics for Rigid Loops,* Programming Research Group - University of Amsterdam, 2006.

[PRG0603] J.A. Bergstra and I. Bethke, *Predictable and Reliable Program Code: Virtual Machine-based Projection Semantics (submitted for inclusion in the Handbook of Network and Systems Administration),* Programming Research Group - University of Amsterdam, 2006.

[PRG0602] J.A. Bergstra and A. Ponse, *Program Algebra with Repeat Instruction,* Programming Research Group - University of Amsterdam, 2006.

[PRG0601] J.A. Bergstra and A. Ponse, *Interface Groups for Analytic Execution Architectures,* Programming Research Group - University of Amsterdam, 2006.

[PRG0505] B. Diertens, *Software (Re-)Engineering with PSF,* Programming Research Group - University of Amsterdam, 2005.

[PRG0504] P.H. Rodenburg, *Piecewise Initial Algebra Semantics,* Programming Research Group - University of Amsterdam, 2005.

[PRG0503] T.D. Vu, *Metric Denotational Semantics for BPPA,* Programming Research Group - University of Amsterdam, 2005.

[PRG0502] J.A. Bergstra, I. Bethke, and A. Ponse, *Decision Problems for Pushdown Threads,* Programming Research Group - University of Amsterdam, 2005.

[PRG0501] J.A. Bergstra and A. Ponse, *A Bypass of Cohen's Impossibility Result,* Programming Research Group - University of Amsterdam, 2005.

[PRG0405] J.A. Bergstra and I. Bethke, *An Upper Bound for the Equational Specification of Finite State Services,* Programming Research Group - University of Amsterdam, 2004.

[PRG0404] J.A. Bergstra and C.A. Middelburg, *Thread Algebra for Strategic Interleaving,* Programming Research Group - University of Amsterdam, 2004.

[PRG0403] B. Diertens, *A Compiler-projection from PGLEc.MSPio to Parrot,* Programming Research Group - University of Amsterdam, 2004.

[PRG0402] J.A. Bergstra and I. Bethke, *Linear Projective Program Syntax,* Programming Research Group - University of Amsterdam, 2004.

[PRG0401] B. Diertens, *Molecular Scripting Primitives,* Programming Research Group - University of Amsterdam, 2004.

[PRG0302] B. Diertens, *A Toolset for PGA,* Programming Research Group - University of Amsterdam, 2003.

[PRG0301] J.A. Bergstra and P. Walters, *Projection Semantics for Multi-File Programs,* Programming Research Group - University of Amsterdam, 2003.

[PRG0201] I. Bethke and P. Walters, *Molecule-oriented Java Programs for Cyclic Sequences,* Programming Research Group - University of Amsterdam, 2002.

The above reports and more are available through the website: www.science.uva.nl/research/prog/

Electronic Report Series

Programming Research Group
Faculty of Science
University of Amsterdam

Kruislaan 403
1098 SJ Amsterdam
the Netherlands

www.science.uva.nl/research/prog/