On Knowledge Spaces and Item Testing

Immanuel Albrecht and Hermann Körndle

Institute of Educational and Developmental Psychology, Technische Universität Dresden, Dresden, Germany
{immanuel.albrecht, hermann.koerndle}@tu-dresden.de

Abstract. First, we briefly introduce some of the fundamental notions of knowledge space theory and how they relate to formal concept analysis. Knowledge space theory has a probabilistic extension which allows it to be utilized in order to assess knowledge states by looking at responses to a variety of test items, which are designed to demand performing different sets of cognitive operations. Second, we introduce an easy extension to lambda calculus in order to incorporate extra-logical operations. Further we define a weight function on term reductions, which is to be used as a model to calculate item response probabilities for test items after task analysis. We use the new model in order to review the probabilistic extension of knowledge space theory.

Keywords: knowledge spaces, lambda calculus, item testing

1 Knowledge Spaces

1.1 Introduction to Knowledge Space Theory

The notion of knowledge spaces was first introduced by Doignon and Falmange in 1985 as follows:

> « The information regarding a particular field of knowledge is conceptualized as a large, specified set of questions (or problems). The *knowledge state* of an individual with respect to that domain is formalized as the subset of all the questions that this individual is capable of solving. A particularly appealing postulate on the family of all possible knowledge states is that it is closed under arbitrary unions. A family of sets satisfying this condition is called a *knowledge space*. » [2].

Furthermore, there is a tightly connected notion of *learning spaces*, which are knowledge spaces with two additional properties: First, one can achieve every higher level of knowledge relative to one's knowledge state by learning one item at a time in a finite number of steps, and second, having a higher level of knowledge does not prevent learning another item, which could have been learned from a subordinate knowledge state. For the formal definition, see [4] 2.2.1.

Also, there is a probabilistic theory of knowledge spaces, which allows for assessment of a subject's knowledge state by evaluating realistic test item responses, which are prone to measurement errors – so called careless errors and lucky guesses.

1.2 Definitions and Relation to Formal Concept Analysis

Definition 1. *(See [4], 2.1.2 and 2.2.2) A pair* $(Q, \mathcal{K})$ *is called a* knowledge structure*, whenever* Q *is a set, and* $\mathcal{K}$ *is a family of subsets of* Q*, i.e.* $\mathcal{K} \subseteq 2^Q$*, such that* $\emptyset \in \mathcal{K}$ *and* $Q \in \mathcal{K}$*. Furthermore such* $(Q, \mathcal{K})$ *is called a* knowledge space*, if* $\mathcal{K}$ *is also closed under* $\bigcup$*, i.e. if for all families with* $\mathcal{F} \subseteq \mathcal{K}$*,* $\bigcup \mathcal{F} \in \mathcal{K}$*.*

Clearly, if $(Q, \mathcal{K})$ is a knowledge structure, then $\mathcal{K}$ together with the set inclusion form a complete lattice: Let $\mathcal{F} \subseteq \mathcal{K}$, then define the lattice join by $\bigvee \mathcal{F} := \bigcup \mathcal{F}$, which yields an element of $\mathcal{K}$ because $\mathcal{K}$ is closed under arbitrary unions. In order to calculate the lattice meet, we may use the canonical equivalence

$$\bigwedge \mathcal{F} = \bigvee \{K \in \mathcal{K} \mid \forall F \in \mathcal{F} : \ K \subseteq F\}$$

The most obvious interpretation of knowledge spaces in terms of formal contexts would be $(\mathcal{K}, \mathcal{K}, \subseteq)$, but it hides some of the structure of $(Q, \mathcal{K})$ in the sense that you have to look at the elements of the objects (or attributes) in order to reconstruct Q. – Yet this operation is outside the scope of formal concept analysis.

Another obvious link between knowledge structures and formal contexts is a *knowledge context*[1] (A, Q, I) where an individual $a \in A$ incides with an item $q \in Q$ if the individual a is *not* capable of solving the item q [3, p.161].

For knowledge spaces $(Q, \mathcal{K})$, it is possible to look at the context $(\mathcal{K}, Q, \not\ni)$: "The intents of the concepts induced by this knowledge context are the complements of the states in $\mathcal{K}$ with respect to Q." [3, p.163]

Last, there is another obvious link that might seem odd at first, but resembles some nice connection between features of knowledge space theory and features of formal concept analysis.

Definition 2. *(See [4], 3.7.1) Let* $(Q, \mathcal{K})$ *be a knowledge structure. The* precedence relation $\precsim$ *wrt.* $(Q, \mathcal{K})$ *is defined as the binary relation on* Q *where for* $q, r \in Q$

$$q \precsim r \ :\Longleftrightarrow \ \forall K \in \mathcal{K} : \ r \in K \Rightarrow q \in K$$

The precedence relation offers another way to think about the knowledge space $(Q, \mathcal{K})$ in terms of formal concept analysis. The defining equivalence for $q \precsim r$ may be read as attribute implication: if the concept related to q is a subconcept of the concept related to r, clearly all attributes of the latter are supposed attributes of the former, too. This yields the formal context $(Q, \mathcal{K}, \epsilon)$ where

$$\epsilon := \{(q, K) \in Q \times \mathcal{K} \mid q \in K\}$$

Let $(Q, \mathcal{K})$ be some knowledge space. Then for $q \in Q$, wrt. $(Q, \mathcal{K}, \epsilon)$, we get

$$\{q\}' = \mathcal{K}_q := \{K \in \mathcal{K} \mid q \in K\}$$

[1] The term *knowledge context* is used for various notions that involve formal concept analysis and knowledge spaces, though.

Thus for $q, r \in Q$, if $\{q\}' \supseteq \{r\}'$, then $\mathcal{K}_r \subseteq \mathcal{K}_q$, which means that for all $K \in \mathcal{K}$, $r \in K \Rightarrow q \in K$, and so

$$\{r\}' \supseteq \{q\}' \;\Leftrightarrow\; r \precsim q$$

and it is easy to see that the single item extents for two indiscernible elements $q_1, q_2 \in Q$ – i.e. for all $K \in \mathcal{K}$, $q_1 \in K \Leftrightarrow q_2 \in K$ – are the same.

Furthermore, for $X \in \mathcal{K}$, since $X \subseteq Q$, we see that the intent of a knowledge state viewed as a set of objects is the principal filter of that state wrt. $(\mathcal{K}, \subseteq)$:

$$X' = \{K \in \mathcal{K} \mid \forall r \in X : \; r \in K\} = \{K \in \mathcal{K} \mid X \subseteq K\} = \uparrow_{(\mathcal{K},\subseteq)} X$$

And we see further, that $X \in \mathcal{K}$ is a concept extent, i.e. $X = X''$:

$$X'' = \{q \in Q \mid \forall K \in X' : \; q \in K\} = \bigcap \uparrow_{(\mathcal{K},\subseteq)} X = X$$

So we see that knowledge states $X \in \mathcal{K}$ correspond to concepts (X, X') of $\mathcal{B}(Q, \mathcal{K}, \epsilon)$.

Now, consider any extent $X \subseteq Q$, i.e. $X'' = X$. In this case, we know by analogous arguments, that for $K \in X'$, also $\uparrow_{(\mathcal{K},\subseteq)} K \subseteq X'$, but we cannot infer that X' is a principal filter wrt. $(\mathcal{K}, \subseteq)$, as this short example demonstrates:

Example 1. Let $Q = \{a, b, c\}$ and $\mathcal{K} = \{\emptyset, \{a\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, Q\}$. Since $\mathcal{K}$ is closed under union, $(Q, \mathcal{K})$ is a knowledge space.

$$\{b\}'' = \{\{a, b\}, \{b, c\}, Q\}' = \{b\} \notin \mathcal{K}$$

We see that $\{b\}'$ is not a principal filter wrt. $(\mathcal{K}, \subseteq)$, because the meet of $\mathcal{K}$ viewed as complete lattice is incompatible with the set meet that is involved in the attribute derivation operator of $(Q, \mathcal{K}, \epsilon)$.

This situation usually arises, when performing a task requires the ability to perform at least one of several distinct subtasks: For instance, if a means that a student knows how to draw a circle with a pen using the left hand, and c means that a student knows how to draw a circle using the right hand, then b could be the ability to write the letter "o" in cursive. In this case, you cannot say that b requires a, or that b requires c; and thus there is an abstract concept $(\{b\}, \{b\}')$ in the concept lattice $\mathcal{B}(Q, \mathcal{K}, \epsilon)$, which does not correspond to a measurable knowledge state with regard to the test items.

To sum it up, we may view $\mathcal{K}$ as a complete sub join-semi-lattice of $\mathcal{B}(Q, \mathcal{K}, \epsilon)$ with the possibility that in some cases $\mathcal{K}$ may or may not be a complete sub lattice of $\mathcal{B}(Q, \mathcal{K}, \epsilon)$.

1.3 Probabilistic Extension

First, we want introduce the general framework which is needed to establish probabilistic methods for knowledge space theory.

Definition 3. *(See [4], 11.1.2) A triple $(Q, \mathcal{K}, p)$ is called* probabilistic knowledge structure*, if $(Q, \mathcal{K})$ is a partial knowledge structure, where Q and $\mathcal{K}$ are finite, and if $p\colon \mathcal{K} \to [0, 1]$ is a probability distribution on $\mathcal{K}$, i.e. $\sum_{K \in \mathcal{K}} p(K) = 1$.*

Definition 4. *(See [4], 11.1.2) A quadruple* $(Q, \mathcal{K}, p, r)$ *is called* basic probabilistic model*, if* $(Q, \mathcal{K}, p)$ *is a probabilistic knowledge structure, and if* r *is a map* $r\colon 2^Q \times \mathcal{K} \to [0,1]$ *– called the* response function *– such that for all* $K \in \mathcal{K}$*,* $r(\cdot, K)$ *is a probability distribution on* 2^Q*, i.e.* $\sum_{R \subseteq Q} r(R, K) = 1$*.*

Local Independence The measurement of a test subject's knowledge state is most easy, if the probability that the test item response does not correctly reflect the subjects knowledge state was only dependent on the item $q \in Q$ and whether the subjects state K contains q or not.

Definition 5. *(See [4], 11.1.2) Let* $(Q, \mathcal{K}, p, r)$ *be a basic probabilistic model. This model satisfies* local independence*, if there are vectors* $\beta, \eta \in \mathbb{R}^Q$*, such that for all* $R \subseteq Q$*,*

$$r(R,K) = \left(\prod_{R \not\ni q \in K} \beta(q)\right)\cdot\left(\prod_{R \ni q \in K} (1-\beta(q))\right)\cdot\left(\prod_{R \ni q \notin K} \eta(q)\right)\cdot\left(\prod_{R \not\ni q \notin K} (1-\eta(q))\right)$$

Clearly, local independence means that careless error probability $\beta(q)$ wrt. to a test item q is the same even for two subjects with huge differences between their respective knowledge states, which may not be appropriate in all situations, yet it dramatically reduces the amount of parameters involved and seems reasonable if both error probabilities are always small. The fact that the lucky guess probability $\eta(q)$ is the same for these two subjects is even less of a problem, since guessing probabilities usually can be reduced to negligibility by appropriate test item design ([4], Remark 11.1.3 (b)). With local independence, one can easily employ a standard χ^2 test on the ratio of the maximum likelihood estimations in order to verify, whether the negligibility of guessing is achieved. If this is the case, the number of free parameters to deal with effectively halves in turn.

Learning Spaces

Definition 6. *Let* $(Q, \mathcal{K})$ *be a knowledge space.* $(Q, \mathcal{K})$ *is called a* learning space*, if for all* $K, L \in \mathcal{K}$ *with* $K \subseteq L$*, there are* $n \in \mathbb{N}$*, and* $q_1, \ldots, q_n \in Q$*, such that for all* $i \in \{1, \ldots, n\}$ *there is a state* $K \cup \{q_1, \ldots, q_i\} \in \mathcal{K}$*; and such that* $K \cup \{q_1, \ldots, q_n\} = L$*. In this case we know that* Q *and* $\mathcal{K}$ *are finite.*

The basic principle of the probabilistic extensions of learning in knowledge structures is stated as follows:

> « The probability that, at the time of the test, a subject is in a state K of the structure is expressed as the probability that this subject *(i)* has successively mastered all the items in the state K, and *(ii)* has failed to master any item immediately accessible from K. » [4, p.198]

This principle may be employed for assessing the knowledge state of some test subject by looking at the subject's responses to a series of test items. The details

of such a procedure are given in *Knowledge Spaces: Applications in Education* [3, pp.140-145] and another procedure involving Markov chains is given in *Learning Spaces* [4]. For the sake of brevity we will give only a quick informal description of the *assessment algorithm* found in [3]:

The assessment algorithm is based on a stochastic process defined by a sequence of random probability distributions L_n of the subject being in a certain state $K \in \mathcal{K}$, a sequence of random variables Q_n that designate which test item $q \in Q$ is asked in the corresponding trial, and a sequence of random variables R_n that yield 1, if the subject's response in the trial was correct and 0 otherwise. The process starts with some initial distribution L_1, the probability of a certain test item being asked in a trial depends on the history of the previous trials and the probability distribution of that trial. The probability of the correct response in a trial depends on the question $q \in Q$ asked, the history of the previous trials and the knowledge state distribution for that trial. The correct response probability is deemed to be $1 - \beta_q$ if the test item is mastered in the subject's latent state. This fact may be used to construct the assessed knowledge state of the student, as the process "$L_n(K_0)$ almost surely converges to 1" for the latent state K_0 of the subject [3, p.143].

2 λ-μ-Calculus

Knowledge and learning space theory defines knowledge states such that they are determined by the ability of a test subject to solve a test item. Therefore we need to investigate how subjects solve test items, or how solution candidates for test items may be constructed. This investigation benefits from a formal apparatus of constructions and operations that is as general as possible. In *Item Construction and Psychometric Models* [9] Tatsuoka argues that according to Glaser [5] achievement tests must reflect the underlying cognitive processes of problem solving, dynamic changes in strategies, and the structure of knowledge and cognitive skills:

> « The correct response to the item is determined by whether all cognitive tasks involved in the item can be answered correctly. Therefore the hypothesis would be that if any of the tasks would be wrong, then there would be a high probability that the final answer would also be wrong. » [9, p.108]

Tatsuoka suggests further, that properties and relations among microlevel and invisible tasks should be explored and predicted [9], which "involves a painstaking and detailed task analysis in which goals, subgoals, and various solution paths are identified in a procedural network (or a flow chart). This process of uncovering all possible combinations of subtasks at the microlevel is essential for making a tutoring system perform the role of the master teacher [...]" [9].

> « Identifying subcomponents of tasks in a given problem-solving domain and abstracting their attributes is still an art. It is necessary that

the process can be made automatic and objective. However, we here assume [...] that any task in the domain can be expressed by a combination of cognitively relevant prime subcomponents. » [9, p.110]

We conclude that the formal apparatus for problem solving must have some way to express extra-logical operations – which may be interpreted as the cognitively relevant prime subcomponents of tasks in a given domain – and it must have some way to express composed tasks and combinations of subtasks in great generality. Our choice of the following extra-logically extended typed lambda calculus as the formalization framework for test item analysis is motived as follows: First, typed lambda calculus typically comes with an algorithm that allows to check whether a given (untyped) lambda term is typable or not, which may be seen as a very rough plausibility test of solution path candidates. Second, types gracefully govern the input and output domains of prime and compound operations. Third, complex task solutions do not have a strict order in which subtasks have to be carried out, this corresponds to different reductions starting from the same term. And last, a formal notion of extra-logical reductions may be interpreted multifariously, for instance as invoking a random process that leads to either a correct or an incorrect solution, as a skill requirement for a given solution path, or as a step-by-step instruction, guideline, hint, etc.

2.1 λ-Calculus

First, we want to fix some definitions regarding the lambda calculus in order to have something precise to refer to, but at the same time we want to point out that our extension of the lambda calculus is quite natural and does not depend on a specific way of formalization. As a general framework for different possible lambda calculi, we use Pure Type Systems in a presentation found in Kamareddine, Laan, and Nederpelt [7], which originates from Berardi [1] and Terlouw [10].

Definition 7. *(See [7], 4.16) Let $\mathbb{V}$ and $\mathbb{C}$ be disjoint sets, that do not contain any of these symbols: "λ", "Π", "(", ")", ".", and ":". The set of* terms *wrt. $\mathbb{V}$ and $\mathbb{C}$ is denoted by $\mathcal{T}(\mathbb{V},\mathbb{C})$. It is defined to be the smallest subset of the support set $|\langle \mathbb{V}\cup\mathbb{C}\cup\{\lambda,\Pi,(,),.,:\}\rangle|$ of the free monoid[2] generated by $\mathbb{V}\cup\mathbb{C}\cup\{\lambda,\Pi,(,),.,:\}$, such that $\mathbb{V}\cup\mathbb{C}\subseteq\mathcal{T}(\mathbb{V},\mathbb{C})$ and for all $A,B\in\mathcal{T}(\mathbb{V},\mathbb{C})$ and all $x\in\mathbb{V}$:*

$$((A)(B))\in\mathcal{T}(\mathbb{V},\mathbb{C}),\ (\lambda v:A.B)\in\mathcal{T}(\mathbb{V},\mathbb{C}),\ \textit{and}\ (\Pi v:A.B)\in\mathcal{T}(\mathbb{V},\mathbb{C})$$

As a notational convention, we may drop parentheses, if they can be restored by successively adding parentheses, where the "("-parentheses are added at the left-most possible positions, and ")"-parentheses are added at the right-most possible positions for λ and Π terms, and at the left-most possible position for application terms: For instance, we may write $\lambda x:A.\lambda y:B.C$ to denote the term

[2] The operation of the free monoid is denoted by juxtaposition, and the neutral element is denoted by ε.

$(\lambda x : A.(\lambda y : B.C))$, *and we may write* xyz *to denote* $((((x)(y)))(z))$, *where* $A, B, C \in \mathcal{T}(\mathbb{V}, \mathbb{C})$ *and* $x, y, z \in \mathbb{V}$.

Definition 8. *Let* $\mathbb{V}$ *and* $\mathbb{C}$ *be sets that satisfy the conditions in Def. 7, and let* $A, X \in \mathcal{T}(\mathbb{V}, \mathbb{C})$ *and* $x \in \mathbb{V}$. *The* substitution *of* x *in* A *by* X *is denoted by* $A[x := X]$. *A formal definition can be found in [6] 1A7 on page 3.*[3] *Since the concept of variable substitution is quite natural to any mathematics, here we give only an informal definition: in* A, *we replace every occurrence of* x *with the word* X, *unless it is part of* B *of a term sub word*[4] *of the form* $(\lambda x : A.B)$ *or* $(\Pi x : A.B)$, *i.e. "*λ*" and "*Π*" bind variables right of ".". For instance*

$$((x)((\lambda x : x.x)))[x := ((y)(z))] = ((((y)(z)))((\lambda x : ((y)(z)).x)))$$

Definition 9. *(See [7], 4.13) Let* $\mathbb{V}$ *and* $\mathbb{C}$ *be sets that satisfy the conditions in Def. 7, and let* $\rightarrow \subseteq \mathcal{T}(\mathbb{V}, \mathbb{C}) \times \mathcal{T}(\mathbb{V}, \mathbb{C})$ *a binary relation on terms.* $\rightarrow$ *is called* compatibility, *if for all* $A, A', B \in \mathcal{T}(\mathbb{V}, \mathbb{C})$ *with* $A \rightarrow A'$, *also the following holds:* $(A)B \rightarrow (A')B$, $(B)A \rightarrow (B)A'$, $\lambda x : A.B \rightarrow \lambda x : A'.B$, $\lambda x : B.A \rightarrow \lambda x : B.A'$, $\Pi x : A.B \rightarrow \Pi x : A'.B$, *and* $\Pi x : B.A \rightarrow \Pi x : B.A'$.

Definition 10. *(See [7], 4.13) Let* $\mathbb{V}$ *and* $\mathbb{C}$ *be sets that satisfy the conditions in Def. 7. The* β-reduction relation *wrt.* $(\mathbb{V}, \mathbb{C})$ *is the smallest compatibility on* $\mathcal{T}(\mathbb{V}, \mathbb{C})$, *denoted by* $\rightarrow_\beta$, *such that* $\triangleright_\beta \subseteq \rightarrow_\beta$; *where* $\triangleright_\beta \subseteq \mathcal{T}(\mathbb{V}, \mathbb{C}) \times \mathcal{T}(\mathbb{V}, \mathbb{C})$ *such that for all* $A, B, C \in \mathcal{T}(\mathbb{V}, \mathbb{C})$ *and* $x \in \mathbb{V}$

$$(((\lambda x : A.B))(C)) \quad \triangleright_\beta \quad B[x := C]$$

The reflexive and transitive closure of $\rightarrow_\beta$ *is denoted by* $\twoheadrightarrow_\beta$, *the reflexive, transitive and symmetric closure of* $\rightarrow_\beta$ *is denoted by* $=_\beta$.

Remark 1. The notion of β-reduction (see Definition 11) is closely related to the notion of α-conversion, which is a compatibility and equivalence relation on $\mathcal{T}(\mathbb{V}, \mathbb{C})$, such that

$$(\lambda x : A.B) \equiv_\alpha (\lambda y : A.B[x := y]) \text{ and } (\Pi x : A.B) \equiv_\alpha (\Pi y : A.B[x := y])$$

if the variable y is not free in the left-hand term. This means that you may rename bound variables, unless you would capture a free variable with the new name. You may read the rest of this paper either by thinking of terms as terms or as $\equiv_\alpha$-equivalence classes. This is a standard feature of lambda calculus.

Definition 11. *Let* $\mathbb{V}$ *and* $\mathbb{C}$ *be sets that satisfy the conditions in Def. 7, let* $\triangleright_x$ *be a binary relation on* $\mathcal{T}(\mathbb{V}, \mathbb{C})$ *and* $\rightarrow_x$ *be the smallest compatibility with* $\triangleright_x \subseteq \rightarrow_x$; *further let* $\underline{n} = \{1, \ldots, n\} \subseteq \mathbb{N}$. *A map* $r \colon \underline{n} \rightarrow \mathcal{T}(\mathbb{V}, \mathbb{C})$ *is a* finite

[3] Or see [7] 4.12, but beware of missing $x \not\equiv y$ and typos in 4.8 (swap A_1 and A_2 on one side).

[4] A term sub word of a term A is a term B, such that there are elements of the free monoid $C, D \in |\langle \mathbb{V} \cup \mathbb{C} \cup \{\lambda, \Pi, (,), ., : \}\rangle|$ with $CBD = A$.

$\triangleright_x$-reduction, *if for all* $i \in \underline{n-1} = \underline{n}\backslash\{n\}$, $r(i) \to_x r(i+1)$; *and if there are maps*

$$\mathrm{pre}, \mathrm{post}\colon \underline{n-1} \to |\langle \mathbb{V} \cup \mathbb{C} \cup \{\lambda, \Pi, (,), ., :\}\rangle|\,, \quad \mathrm{redex}\colon \underline{n-1} \to \mathcal{T}(\mathbb{V}, \mathbb{C})$$

such that for all $i \in \underline{n-1}$ *:*

$$\mathrm{pre}(i)\mathrm{redex}(i)\mathrm{post}(i) = r(i) \;\; \textit{and}$$

$$\exists t_i \in \mathcal{T}(\mathbb{V}, \mathbb{C}) : \; \mathrm{redex}(i) \triangleright_x t_i \;\; \textit{such that} \;\; \mathrm{pre}(i)t_i\mathrm{post}(i) = r(i+1)$$

The set of all finite $\triangleright_x$*-reductions is denoted by* ∇_x*, its subset of all finite* $\triangleright_x$*-reductions with* $r(1) = t$ *for* $t \in \mathcal{T}(\mathbb{V}, \mathbb{C})$ *is denoted by* ∇_x^t.

Definition 12. *(See [7], 4.18) A tuple* $(\mathbb{V}, \mathbb{C}, \mathbf{S}, \mathbf{A}, \mathbf{R})$ *is called* pure type system specification, *if* $\mathbb{V}$ *and* $\mathbb{C}$ *are sets that satisfy the conditions in Def. 7; and if* $\mathbf{S} \subseteq \mathbb{C}$, $\mathbf{A} \subseteq \mathbf{S} \times \mathbf{S}$, *and* $\mathbf{R} \subseteq \mathbf{S} \times \mathbf{S} \times \mathbf{S}$. *In this context, we call* $\mathbf{S}$ *the set of sorts,* $\mathbf{A}$ *the set of axioms, and* $\mathbf{R}$ *the set of* Π*-formation rules.*

These definitions are sufficient for our purposes. For a complete introduction to lambda calculus and pure type systems, we refer you to [7], sections 4a through 4c.

2.2 μ-Extension

In this section, we want to define a generic extension of lambda calculus that allows to deal with extra-logical operations. Since these operations are not part of the lambda calculus, we need to specify a set of symbols which are regarded as new constants from the point of some given lambda calculus, and we need to specify a set of new derivation rules for typed terms that govern the correct formal types of these symbols. Part of this work can be done by altering the pure type system specification of the underlying lambda calculus, part of this work has to be done by hand.

Definition 13. *The* μ-extension alphabet *is defined to be the set* $\mathbb{M}$ *that contains the distinct symbols "[", "]", ";", "*μ_i*", "*ν_i*", and "*m_i*" for all* $i \in \mathbb{N}$;[5] *i.e.*

$$\mathbb{M} = \{[, ;,]\} \cup \{\mu_i, \nu_i, m_i \mid i \in \mathbb{N}\}$$

The set of ν-constants $\mathcal{M}_\nu$ *is defined to be the smallest subset of* $|\langle \mathbb{M} \rangle|$, *that has the following properties: For all* $i \in \mathbb{N}$, $m_i \in \mathcal{M}_\nu$; *and for all* $i, k \in \mathbb{N}$, *if* $x_1, \ldots, x_k \in \mathcal{M}_\nu$, *then* $\nu_i[x_1; \ldots; x_k] \in \mathcal{M}_\nu$.

The set of μ-constants $\mathcal{M}$ *is defined as* $\mathcal{M} := \{\mu_i \mid i \in \mathbb{N}\} \cup \mathcal{M}_\nu$.

Definition 14. *A pair* $(\mathbf{S}, S_0, M, a, p, v)$ *is called* μ-specification, *if* $S_0 \in \mathbf{S}$, *and if* M, a, p, *and* v *are maps, such that* $M\colon \mathbb{N} \to \mathbf{S}$, $a\colon \mathbb{N} \to \mathbb{N}$, $p\colon \mathbb{N} \to \mathbf{S}^{(\mathbb{N})}$, $v\colon \mathbb{N} \to \mathbf{S}$, *and such that for all* $i \in \mathbb{N}$, $p(i) \in \mathbf{S}^{a(i)}$. *Here,* $M(i)$ *specifies the type of the constant* m_i; $a(i)$ *specifies the arity of the symbol* μ_i, *whereas* $p(i)$ *specifies the parameter types of* μ_i *and* $v(i)$ *specifies the value type of* μ_i.

[5] Of course you could use a finite subset of $\mathbb{N}$ as well.

For instance, each of the symbols μ_i may encode one of the operations listed in Table 1 in [8] with appropriate $a(i)$ and $p(i)$.

Definition 15. *Let* $(\mathbb{V}, \mathbb{C}, \mathbf{S}, \mathbf{A}, \mathbf{R})$ *be a pure type system specification, such that* $\mathcal{M} \cap \mathcal{T}(\mathbb{V}, \mathbb{C}) = \emptyset$*; and let* $(\mathbf{S}', S_0, M, a, p, v)$ *be a* μ*-specification, such that* $\mathbf{S}' \subseteq \mathbb{C}$*. The* μ-extension *of* $(\mathbb{V}, \mathbb{C}, \mathbf{S}, \mathbf{A}, \mathbf{R})$ *wrt.* $(\mathbf{S}', S_0, M, a, p, v)$ *is defined to be the tuple* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$*, where* $\mathbb{C}_\mu := \mathcal{M} \cup \mathbb{C}$,[6] $\mathbf{S}_\mu := \mathbf{S} \cup \mathbf{S}'$ *and*

$$\mathbf{A}_\mu := \mathbf{A} \cup \{(m_i, M(i)), (M(i), S_0), (v(i), S_0) \mid i \in \mathbb{N}\}$$

A tuple $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *that is a* μ*-extension of a pure type system specification wrt. some* μ*-specification is called* λ-μ-specification *from here on.*

For technical reasons, we cannot express the axioms for the correct function type of μ_i ($i \in \mathbb{N}$) by elements of the set $\mathbf{A}_\mu$, since μ_i may have a compound type[7] that is represented by a term from the set $\mathcal{T}(\mathbb{V}, \mathbb{C}_\mu) \backslash \mathbb{C}_\mu$. Furthermore, ν_i involves a different derivation rule that takes care of the input and output types of the μ_i operation. Therefore we need to define three and a half additional derivation rule schemes for the λ-μ-calculus.

Definition 16. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ*-specification.*

The derivation rules of the corresponding λ-μ-calculus *are the derivation rules of the pure type system corresponding to* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R})$ *plus the following additional rules for all* $i \in \mathbb{N}$, $x \in \mathbb{V}$, $x_1, \ldots, x_{a(i)} \in \mathcal{M}_\nu$*:*

(axiom**μ_i**) $\quad \langle\rangle \vdash \mu_i : \Pi x : p(i)_1. \ldots \Pi x : p(i)_{a(i)}.v(i)$

(axiom**μ_i**') $\quad \langle\rangle \vdash \Pi x : p(i)_1. \ldots \Pi x : p(i)_{a(i)}.v(i) : S_0$

(appl**μ_i**) $\quad \dfrac{\langle\rangle \vdash x_1 : p(i)_1 \quad \ldots \quad \langle\rangle \vdash x_{a(i)} : p(i)_{a(i)}}{\langle\rangle \vdash \mu_i x_1 \ldots x_{a(i)} : v(i)}$

(appl**ν_i**) $\quad \dfrac{\langle\rangle \vdash x_1 : p(i)_1 \quad \ldots \quad \langle\rangle \vdash x_{a(i)} : p(i)_{a(i)}}{\langle\rangle \vdash \nu_i[x_1; \ldots; x_{a(i)}] : v(i)}$

A term $t \in \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$ *that may be derived using the above derivation rules is called* typable *wrt. the* λ-μ*-specification.*

Remark 2. The derivation rule **(appl**μ_i**)** is unnecessary from a purist point of view, since it may be expressed using $a(i)$ subsequent application rules **(appl)**. Yet this does not treat μ_i as function with multiple parameters which are applied at once, but as a Curry-transformed version that maps its single parameter to another single parameter function, to which then the next parameter is applied

[6] If $\mathbb{C}$ was finite or countably infinite, $\mathbb{C}_\mu$ is also countably infinite; thus we do not break any countable infinity assumptions on $\mathbb{V}$ and $\mathbb{C}$, as made on p.112 in [7].

[7] This is the case, whenever $a(i) \neq 0$.

and so on. Since prime subcomponents of tasks should be indivisible into subtasks, the partial applications of formal operations appear to be purely logic. In order to reflect the intuition of prime formal operations with multiple parameters, we introduce a rule scheme that allows application of all parameters of formal operations in one step.

For instance, a μ-reduction steps may correspond to performance of operations as given by a line in Table 4 [8].

Definition 17. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ-*specification. The* μ-reduction relation *wrt. that* λ-μ-*specification is the smallest compatibility on* $\mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$ *– denoted by* $\rightarrow_\mu$ *– with* $\triangleright_\mu \subseteq \rightarrow_\mu$*, where* $\triangleright_\mu \subseteq \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu) \times \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$ *such that for all* $i \in \mathbb{N}$*,* $x_1, \ldots, x_{a(i)} \in \mathcal{M}_\nu$*,*

$$\underbrace{(\ldots(}_{2 \cdot a(i) \times \text{“(”}} \mu_i)(x_1)) \underbrace{\ldots}_{\substack{\text{“})(x_i))\text{” for} \\ 1 < i < a(i)}} (x_{a(i)})) \quad \triangleright_\mu \quad \nu_i[x_1; \ldots; x_{a(i)}]$$

This defines the notion of finite $\triangleright_\mu$-reductions (see Definition 11), that reduce the formal operations μ_i ($i \in \mathbb{N}$), which are applied to the extra-logical value constants $x_1, \ldots, x_{a(i)} \in \mathcal{M}_\nu$, to their canonical result $\nu_i[x_1; \ldots; x_{a(i)}] \in \mathcal{M}_\nu$.

We now have to check that this reduction works in a way, such that if a term $T \in \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$ is well-typed wrt. the λ-μ-specification, then also all terms $R \in \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$ with $T \rightarrow_\mu R$ are well-typed wrt. that λ-μ-specification. Since μ_i has the type $\Pi x : p(i)_1. \ldots \Pi x : p(i)_{a(i)}.v(i)$, and if $x_i, \ldots, x_{a(i)}$ are terms such that x_j has the type $p(i)_j$ for $j \in \{1, \ldots, a(i)\}$, the term $\mu_i x_1 \ldots x_{a(i)}$ is well-typed and has the type $v(i)$. This means, that if we have a derivation for the type of some term $T \in \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$, i.e. if T is well-typed, we can obtain a derivation for every term $R \in \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$ with $T \rightarrow_\mu R$ by replacing the appropriate μ_i-subterms with the appropriate $\nu_i[\ldots]$-subterms, and then using the derivation rule **(appl**ν_i**)** as a replacement for **(appl**μ_i**)**[8].

2.3 Stateful μ-Actions

In this section, we give the common abstraction that will help interpreting finite $\triangleright_\mu$-reductions as operations performed one after another.

Definition 18. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ-*specification. A triple* $(X, A, \mathbb{A})$ *is called* stateful μ-action*, if* X *is a set – called the set of* subject states*;* A *is a set – called the set of* auxiliary states*, and if*

$$\mathbb{A} \colon X \times \mathbb{N} \rightarrow (X \times A)^{A^{(\mathbb{N})} \times \mathcal{M}_\nu}$$

is a map, such that for all $i \in \mathbb{N}$ *and* $x \in X$*; the map* $\mathbb{A}(x, i)$ *– called the* μ_i-action for x *– has the following signature:*

$$\mathbb{A}(x, i) \colon A^{a(i)} \times \mathcal{M}_\nu \rightarrow X \times A$$

[8] or the $a(i)$ usages of **(appl)**

Definition 19. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ-*specification, and* $\mathcal{A} = (X, A, \mathbb{A})$ *be a stateful* μ-*action. Furthermore, let* $n \in \mathbb{N}$, $r \in \nabla_\mu$ *be a finite* $\triangleright_\mu$-*reduction,*

$$\mathrm{redex}\colon \underline{n-1} \to \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$$

and

$$\mathrm{pre}\colon \underline{n-1} \to |\langle \mathbb{V} \cup \mathbb{C}_\mu \cup \{\lambda, \Pi, (,), ., :\}\rangle|$$

be the corresponding functions as in Definition 11. The triple $(r, \mathrm{redex}, \mathrm{pre})$ *is a* solution strategy*, if the last term is a constant symbol, i.e.* $r(n) \in \mathcal{M}_\nu$*; and if the first term* $r(1)$ *is typable wrt. the* λ-μ-*specification and consists only of symbols from* $\{\mu_i, m_i \mid i \in \mathbb{N}\} \cup \{(,)\}$*, i.e.*

$$r(1) \in |\langle \{\mu_i, m_i \mid i \in \mathbb{N}\} \cup \{(,)\}\rangle|$$

Since redex *and* pre *are canonical for* r*, we may denote the solution strategy just by* r*. The set of all solution strategies is denoted by* $\diamondsuit_\mu$*.*

Definition 20. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ-*specification, and* $\mathcal{A} = (X, A, \mathbb{A})$ *be a stateful* μ-*action and* $(r, \mathrm{redex}, \mathrm{pre})$ *be a solution strategy. Then the* performance map of r *wrt.* $\mathcal{A}$ *and the* λ-μ-*specification*

$$\begin{aligned} r^{\mathcal{A}} :& X \times A^{\mathbb{N}} \to X \times A, \\ & (x, (\alpha_i)_{i\in\mathbb{N}}) \mapsto \left(r^{\mathcal{A}}_X(x, (\alpha_i)_{i\in\mathbb{N}}), r^{\mathcal{A}}_A(x, (\alpha_i)_{i\in\mathbb{N}})\right) \end{aligned}$$

is defined by the sequence of $\triangleright_\mu$*-reduction steps of* r*:*

Let $x \in X$ *and* $(\alpha_i)_{i\in\mathbb{N}} \in A^{\mathbb{N}}$*, and let* $i \in \{1, \ldots, n\}$ *be the running index for the rest of this definition. We define the map*

$$\bar{r}_i\colon \{1, \ldots, k_i\} \to \mathbb{V} \cup \mathbb{C}_\mu \cup \{\lambda, \Pi, (,), ., :\}$$

to be the map such that $k_i \in \mathbb{N}$ *and* $\bar{r}_i(1)\bar{r}_i(2)\ldots\bar{r}_i(k_i) = r(i)$ *wrt. the freely generated monoid* $\langle \mathbb{V} \cup \mathbb{C}_\mu \cup \{\lambda, \Pi, (,), ., :\}\rangle$*, i.e.* $\bar{r}_i$ *is the symbol-at-index map of the term* $r(i)$ *viewed as a word. Further let* $l_i \in \mathbb{N}$ *be the length of the word* $\mathrm{pre}(i)$*, i.e. the number such that there are* $\sigma_1, \ldots, \sigma_{l_i} \in \mathbb{V} \cup \mathbb{C}_\mu \cup \{\lambda, \Pi, (,), ., :\}$ *with* $\mathrm{pre}(i) = \sigma_1\sigma_2\ldots\sigma_{l_i}$*. We define the auxiliary maps* $\bar{X}\colon \underline{n} \to X$ *and, for* $i \in \{1, \ldots, n\}$*,* $A_i\colon \{1, \ldots, k_i\} \to A$*: We set* $\bar{X}(1) = x$*, and*

$$A_1(j) = \begin{cases} \alpha_h & \text{if } \bar{r}_i(j) = m_h \text{ for } h \in \mathbb{N} \\ \alpha_0 & \text{else} \end{cases}$$

and for $g \in \underline{n-1}$*:*

$$A_{g+1}(j) = \begin{cases} A_g(j) & \text{if } j \le l_g \\ \pi_A(\mathbb{A}(\bar{X}(g), f_g)((A_g(e_{g,1}), \ldots, A_g(e_{g,a(i)})), \bar{r}_{g+1}(l_g+1))) & \text{if } j = l_g + 1 \\ A_g(j + k_g - 1) & \text{if } l_g + 1 < j \end{cases}$$

and

$$\bar{X}(g+1) = \pi_X(\mathbb{A}(\bar{X}(g), f_g)((A_g(e_{g,1}), \ldots, A_g(e_{g,a(i)})), \bar{r}_{g+1}(l_g+1)))$$

where $e_{g,d_g} = 5 \cdot d_g + 2 \cdot a(f_g) - 1$ *for* $d_g \in \{1, \ldots, a(f_g)\}$*, and where* $f_g \in \mathbb{N}$ *such that* $\bar{r}_g(l_g + 1) = \mu_{f_g}$*, whereas* π_X *and* π_A *denote the respective coordinate projections of* $X \times A$*.*

Finally, we set

$$r^{\mathcal{A}}(x, (\alpha_i)_{i \in \mathbb{N}}) = (\bar{X}(n), A_n(1))$$

It is obvious that the above definition requires explanation: Given is a finite $\triangleright_\mu$-reduction r that ends in some result $r(n) \in \mathcal{M}_\nu$ of extra-logical operations, which is a single constant symbol term in $\mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$. Furthermore, the given reduction starts with a well-typed term $r(1)$ that consists only of μ_i and m_i symbols, and the symbols for their respective applications.

We are interested in the state transition corresponding to the operation sequence of r, which is a simultaneous transition of an input subject state and a vector of auxiliary states associated with the extra-logical constants m_i into an output subject state and a single output auxiliary state associated with the result $r(n)$. The reduction r induces an order in which different actions are carried out, and the map $\bar{X}$ represents the state of the subject between the actions. Furthermore the maps A_g represent the auxiliary states of the intermediate results associated with the symbols from $\mathcal{M}_\nu$.[9]

Here, the subject state may be a representation of the current knowledge, skills, short and long term memory, motivation and fatigue of a subject, whereas the auxiliary state may measure the correctness of the intermediate results, partial response times and the amount of effort that was put into solving the subtask. Or the subject state may be a distribution of knowledge states and the auxiliary state may be a distribution of the correctness of the intermediate results; or – slightly abusing the original idea – the subject state may keep track of the required skills, whereas the auxiliary state may keep track of the required effort.

Last, we want to point out, that when given a λ-μ-term that contains no $\nu_i[\ldots]$ symbols, and if that term has a finite $\triangleright_{\beta,\mu} = \triangleright_\beta \cup \triangleright_\mu$-reduction r with $r(n) \in \mathcal{M}_\nu$, we may postpone all $\triangleright_\mu$-reduction steps to the end, since neither μ-redexes nor μ-reducts have any 'λ' symbol, which is part of the β-redex – roughly speaking: a $\triangleright_\mu$-reduction step cannot create new or remove old work for the $\triangleright_\beta$-reduction. This means, that if we have some typable term $t \in \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$, we may calculate its β-normal form before invoking the extra-logical μ-machinery.

2.4 Solution Probabilities for Test Items Formalized by λ-μ-Specifications

Definition 21. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ*-specification. A pair* (m_q, S_q) *is called* test item *wrt. the* λ-μ*-specification, if* m_q *is a* μ*-constant symbol* m_i*, i.e.* $m_q \in \{m_i \mid i \in \mathbb{N}\} \subseteq \mathcal{M}_\nu$*, and if* S_q *is the type of the solution, i.e.* $S_q \in \mathbf{S}_\mu$*.*

The correct type corresponding to (m_q, S_q) *is the type of functions from* $M(i)$ *to* S_q*, i.e.* $\Pi x : M(i).S_q$ *where* $i \in \mathbb{N}$ *such that* $m_q = m_i$*.*

[9] Partial maps A_g fit the situation better, but require more technical details.

Definition 22. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ*-specification, and let* $t \in \mathcal{T}(\mathbb{V}, \mathbb{C}_\mu)$ *be a term. We call* t a solution candidate*, if* t *is typable wrt. the* λ-μ*-specification. In this case,* t_β *denotes the* $\triangleright_\beta$*-normal form of* t*, which then exists (see [7], Theorem 4.40: Strong Normalization Theorem for ECC).*

Definition 23. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ*-specification, let* t *be a solution candidate, and* $q = (m_i, S_q)$ *be a test item wrt. the* λ-μ*-specification, where* $i \in \mathbb{N}$ *and* $m_i \in \mathcal{M}$*. Then* t *is a* solution procedure *for* q*, if* t *does not contain any symbols from* $\mathcal{M}_\nu$ *and has the correct type corresponding to* q*, i.e. if there is a valid derivation tree with the root judgement*

$$\langle\rangle \quad \vdash \quad t \quad : \quad \Pi x : M(i).S_q$$

We denote the set of all solution procedures for q *by* Ξ_q*.*

For instance, every method listed in Table 1 [8] corresponds to such a solution procedure.

Remark 3. Clearly, if $q = (m_i, S_q)$ and $r = (m_j, S_r)$ are test items such that $S_q = S_r$ and $M(i) = M(j)$, then every solution procedure for q is a solution procedure for r and vice versa.

Now consider a test item $q = (m_q, S_q)$ that is given to some test subject. There are two cases: *(i)* the subject does not find a solution procedure for q, or *(ii)* we may view the solution procedure as a discrete random variable ξ which may take values from Ξ_q. In the first case, the probability of giving a correct response equals the probability of a correct guess. In the second case, we may determine the probability of a correct response under the condition, that $\xi = t$ for $t \in \Xi_q$ by utilizing a stateful μ-action:

Definition 24. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ*-specification,* Q *is a set, such that all its elements* $q \in Q$ *are test items wrt. the* λ-μ*-specification. A tuple* $(x_0, \alpha, \gamma, \beta, \eta, \bar{\eta}, \equiv_\nu, \delta)$ *is called* formal test subject *wrt.* Q*, if*

1. $\alpha = (\alpha_i)_{i \in \mathbb{N}} \in [0,1]^{\mathbb{N}}$ *– the probabilities for correctly understanding* m_i*,*
2. $\gamma = (\gamma_q)_{q \in Q} \in [0,1]^Q$ *– the prob. for discovery of a correct solution procedure,*
3. $\bar{\eta} = (\bar{\eta}_q)_{q \in Q} \in [0,1]^Q$ *– the prob. for guessing if no procedures was discovered,*
4. $\beta = (\beta_i)_{i \in \mathbb{N}} \in [0,1]^{\mathbb{N}}$ *– the probabilities for failing to perform* μ_i*,*
5. $\eta = (\eta_i)_{i \in \mathbb{N}} \in [0,1]^{\mathbb{N}}$ *– the prob. for guessing the correct results for* μ_i*,*
6. $\equiv_\nu \subseteq \mathcal{M}_\nu \times \mathcal{M}_\nu$ *is an equivalence relation identifying results obtained in different ways*[10]*,*
7. $\delta = (\delta_x)_{x \in \mathcal{M}_\nu/\equiv_\nu} \in [0,1]^{\mathcal{M}_\nu/\equiv_\nu}$ *– the prob. for keeping the result* x *in memory,*
8. $x_0 \colon \mathcal{M}_\nu/\equiv_\nu \to [0,1] \times [0,1]$ *–* $x_0(x) = (p_1, p_2)$ *means that* x *is taken from memory with prob.* p_1*, and* p_2 *is the prob. that the retrieved result is correct.*

[10] A nice property for $\equiv_\nu$ would be that it only identifies results with that have the same type, but this is not necessary from a formal point of view.

Definition 25. *Let* $(\mathbb{V}, \mathbb{C}_\mu, \mathbf{S}_\mu, \mathbf{A}_\mu, \mathbf{R}, S_0, a, p, v)$ *be a* λ-μ*-specification, and* $s = (x_0, \alpha, \gamma, \beta, \eta, \bar{\eta}, \equiv_\nu, \delta)$ *be a formal test subject. The* stateful μ-action associated with s *is defined to be the triple* $\mathcal{A}_s = (X_s, [0,1], \mathbb{A}_s)$ *where*

$$X_s = ([0,1] \times [0,1])^{\mathcal{M}_\nu/\equiv_\nu}$$

and where

$$\mathbb{A}_s \colon X_s \times \mathbb{N} \to (X_s \times [0,1])^{[0,1]^{(\mathbb{N})} \times \mathcal{M}_\nu}$$

such that for all $x \in X_s$ *and* $i \in \mathbb{N}$*;*

$$\begin{aligned} \mathbb{A}_s(x,i) \colon\ & [0,1]^{a(i)} \times \mathcal{M}_\nu \to X_s \times [0,1], \\ & \left(\left(p_1, \ldots, p_{a(i)}\right), r\right) \mapsto (y, p_c + p_g + p_m) \end{aligned}$$

Where:

$$\begin{aligned} p_c &= \left(\prod_{i=0}^{a(i)} p_i\right) \cdot (1 - \beta_i) \cdot \left(1 - \pi_1\left(x\left([r]_{\equiv_\nu}\right)\right)\right) \\ p_g &= \eta_i \cdot \beta_i \cdot \left(1 - \pi_1\left(x\left([r]_{\equiv_\nu}\right)\right)\right) \\ p_m &= \pi_1\left(x\left([r]_{\equiv_\nu}\right)\right) \cdot \pi_2\left(x\left([r]_{\equiv_\nu}\right)\right) \end{aligned}$$

and

$$\begin{aligned} y \colon\ & \mathcal{M}_\nu/\equiv_\nu \to [0,1] \times [0,1], \\ & t \mapsto \begin{cases} x(t) & \text{if } t \neq [r]_{\equiv_\nu} \\ (\pi_1(x(t)) + (1 - \pi_1(x(t))) \cdot \delta_i, & \\ \qquad p_c + p_g + p_m) & \text{if } t = [r]_{\equiv_\nu} \end{cases} \end{aligned}$$

Here, π_1 *and* π_2 *denote the respective coordinate projections of* $[0,1] \times [0,1]$*; and* $[r]_{\equiv_\nu}$ *denotes the equivalence class of* r *wrt.* $\equiv_\nu$*.*

The above definitions interact in the following way: We assume, that we have a μ-specification $(\mathbf{S}', S_0, M, a, p, v)$ which models the domain of knowledge we are investigating. Each $S \in \mathbf{S}'$ stands for a certain way of purposeful information representation. The symbols $m_i \in \mathcal{M}_\nu$ stand for some information represented in the way of $M(i)$. The symbols μ_i stand for operations that process $a(i)$ pieces of information represented in the ways of $p(i)_1, \ldots p(i)_{a(i)}$ to some piece of information represented in the way of $v(i)$.

A test item is then formalized as some given problem represented by the symbol m_q and a task objective $S_q \in \mathbf{S}'$, where we view the task as re-representing the given information in a certain way that elucidates the answer. In order to solve that item, a test subject has to find a series of μ_i operations that turn M_q-representations into S_q-representations, where $m_q : M_q$. In general, the subject will have to perform general logic operations in order to create a solution strategy, such as using the given information m_q in different ways to perform different operations – and this is where λ-calculus is needed.

Consider that you want solve the question $1+5=?$, and that your operation at hand is 'add two single digit decimal numbers'. Then you would have to extract two different pieces of information: the left and right operands, and your solution candidate for that kind of tasks could be

$$\lambda x : S_{\text{easy Addition}}.\mu_1\ (\mu_2\ x)\ (\mu_3\ x) \quad : \quad \Pi x : S_{\text{easy Addition}}.S_{\text{integerNumber}}$$

where μ_1 represents the addition operation, and where μ_2 and μ_3 represent the operand extraction. Please note that there is no need for logic decisions based on the results of the μ_i operations, since the correct decisions may be encoded by $\mathbf{S}'$: carrying information about two easy numbers together with the purpose 'addition' means inhabiting the type $S_{\text{easy Addition}}$. This way, it is possible to decide whether a solution strategy may work or not based on the type of the representing term alone.

After choosing a solution strategy, the test subject has to perform the operations accordingly. The various sources of errors and lucky guesses – which may be due to actual operation or due to remembering correct or wrong (intermediate) results – give rise to a formal test subject, which corresponds to the test subject and the particular time of testing.

The stateful μ-action that is associated with the formal test subject then acts in the following way: The operations are carried out according to the finite μ-reduction in a probabilistic manner, such that if the result is available from memory, then that result is used, otherwise the result is derived from the input, and in case that the input is correct, there is a probability of introducing a new error, and a probability of guessing the correct result. Then the memory is updated with the new derived result.

We sketch a patch of the assessment algorithm [3, pp.140-145], which may lead to a reduction in free parameters for knowledge domains, if there are less operations than test items – which may be achieved by appropriate design. We give a definition that replaces the response rule axiom [R]:

Definition 26. *Let Q be a set of test items wrt. a λ-μ-specification, $Q \ni q = (m_q, S_q)$, $s = (x_0, \alpha, \gamma, \beta, \eta, \bar{\eta}, \equiv_\nu, \delta)$ a formal test subject wrt. Q; let R be a random variable with outcomes $\{0,1\}$, and ξ be a random variable with outcomes $\Xi_q \cup \{\bot\}$.*

The pair (R,ξ) is called formal response *of the subject s to the test item q, if the following equations are satisfied for all $t \in \Xi_q$: $\mathbb{P}(\xi = \bot) = \gamma_q$ and $\mathbb{P}(R = 1 | \xi = \bot) = \bar{\eta}_q$ and for $\Upsilon := \{(r, \text{redex}, \text{pre}) \in \diamondsuit_\mu \mid r \in \nabla_\mu^{(t\ m_q)_\beta}\}$;*

$$\mathbb{P}(R = 1 | \xi = t) = \frac{1}{\#\Upsilon} \sum_{r \in \Upsilon} r^{\mathcal{A}_s}\ (x_0, \alpha)$$

where $(t\ m_q)_\beta$ denotes the $\triangleright_\beta$-normal form of the application term $(t\ m_q)$, and $\mathcal{A}_s$ the stateful μ-action associated with s.

3 Discussion

Although the assessment algorithm of knowledge space theory may be utilized

to uncover the latent knowledge state of a test subject, it is merely blind to any learning process involved during the trials. The careless error and lucky guess parameters do not allow to investigate relations between the underlying cognitive processes of different test items. We want to investigate those relations, and thus we cannot assume that the process of solving a test item does not involve any state changes – as it is implausible that the same cognitive operation with the same input parameters is repeated within the short timespan of a few solution steps.

The model we introduced implies that expertise available to the test subject may grow steadily just by solving test items, which means that the careless error probabilities for items that belong to basic knowledge states should in fact sink as the subject reaches a higher knowledge state – the probability of careless errors when doing single digit additions should be less for students in senior class compared to when they newly learned it.

In future work, the described models may be adapted to assess a concept associated with some measurement, where the different trials would be the objects of the context and the interdependencies between attributes would be modeled using λ-μ-calculus, leading to a stochastic assessment procedure.

References

1. Berardi, S.: Towards a mathematical analysis of the coquand-huet calculus of constructions and the other systems in barendregt's cube. Tech. rep., Dept. of Computer Science, Carnegie-Mellon University and Dipartimento Matematico, Universita di Torino (1988)
2. Doignon, J.P., Falmagne, J.C.: Spaces for the assessment of knowledge. International Journal of Man-Machine Studies 23(2), 175 – 196 (1985)
3. Falmagne, J., Doble, C., Albert, D., Eppstein, D., Hu, X.: Knowledge Spaces: Applications in Education. Springer-Verlag New York Incorporated (2013)
4. Falmagne, J.C., Doignon, J.P.: Learning Spaces. Springer (2010)
5. Glaser, R.: The integration of instruction and testing. In: Freeman, E. (ed.) The redesign of testing in the 21st century: Proceedings of the 1985 ETS Invitational Conference. pp. 45–58. Princeton, NJ: Educational Testing Service (1985)
6. Hindley, J.R.: Basic simple type theory. Cambridge University Press (1997)
7. Kamareddine, F., Laan, T., Nederpelt, R.: A Modern Perspective on Type Theory: From its Origins until Today. Applied logic series, Kluwer Academic Publishers (2006)
8. Korossy, K.: Modeling Knowledge as Competence and Performance. In: Albert, D., Lukas, J. (eds.) Knowledge Spaces: Theories, Empirical Research, and Applications, pp. 103–132. Mahwah, NJ (1999)
9. Tatsuoka, K.K.: Item Construction and Psychometric Models Appropriate for Constructed Responses. In: Bennett, R., Ward, W. (eds.) Construction Versus Choice in Cognitive Measurement: Issues in Constructed Response, Performance Testing, and Portfolio Assessment, chap. 6, pp. 107–133. Routledge, New York and London (2009)
10. Terlouw, J.: Een nadere bewijstheoretische analyse von gstt's. Tech. rep., Department of Computer Science, University of Nijmegen (1989)