

A multi-modal type theory for representing data accessibility in a network

Giuseppe Primiero

FWO - Flemish Research Foundation Centre for Logic and Philosophy of Science, Ghent University IEG - Oxford University



Giuseppe.Primiero@Ugent.be http://www.philosophy.ugent.be/giuseppeprimiero/

Proof Systems for Program Logics Workshop 10th, July, 2010 - Edinburgh – UK

(ロ) (日) (日) (日) (日)

Outline



A modal contextual type theory with judgemental modalities







3 b 4



2 A modal contextual type theory with judgemental modalities







G. Primiero (Ghent University)

Multi-Modal Type Theory

PSPL2010 3 / 31

.

Image: A matrix

From Constructive Modalities to Modal Type Theories

Semantic modelization

- [Simpson(1994)];
- [Bierman and de Paiva(2000)],[Alechina, Mendler, de Paiva(2001)];
- Lambda5 and MI5 implemented in [Murphy(2008)] and [Murphy, Crary, and Harper(2008)] for Grid Computing;



From Constructive Modalities to Modal Type Theories

Semantic modelization

- [Simpson(1994)];
- [Bierman and de Paiva(2000)],[Alechina, Mendler, de Paiva(2001)];
- Lambda5 and MI5 implemented in [Murphy(2008)] and [Murphy, Crary, and Harper(2008)] for Grid Computing;
- Constructive modalities and modal type theories:
 - [Pfenning and Davies(2001)];
 - [Nanevski, Pfenning, and Pientka(2008)];
 - Reason about distributed and staged computation: [Moody(2003)], [Davies and Pfenning(2001)], [Jia and Walker(2004)].



Resources and Locations via Types

- Type Theories for safe distributed computing
 - [Davies and Pfenning(2001)], [Jia and Walker(2004)]
 - Ability to represent heterogeneity w.r.t. properties, resources, devices, software, services;
 - Locally sound and complete modalities;
 - Type theoretical formulations / Natural Deduction / Sequent Calculi.



Resources and Locations via Types

- Type Theories for safe distributed computing
 - [Davies and Pfenning(2001)], [Jia and Walker(2004)]
 - Ability to represent heterogeneity w.r.t. properties, resources, devices, software, services;
 - Locally sound and complete modalities;
 - > Type theoretical formulations / Natural Deduction / Sequent Calculi.
- A typed λ-calculus with stationary situations and flowing informations
 - [Borghuis, Feijs(2000)]
 - Interesting take on the representation of the order of commands;
 - Inspiring for the focus on recover of data from locations.



Resources and Locations via Types

- Type Theories for safe distributed computing
 - [Davies and Pfenning(2001)], [Jia and Walker(2004)]
 - Ability to represent heterogeneity w.r.t. properties, resources, devices, software, services;
 - Locally sound and complete modalities;
 - > Type theoretical formulations / Natural Deduction / Sequent Calculi.
- A typed λ-calculus with stationary situations and flowing informations
 - [Borghuis, Feijs(2000)]
 - Interesting take on the representation of the order of commands;
 - Inspiring for the focus on recover of data from locations.
- A modal logic for local resources
 - [Park(2006)]
 - Problem of distinguishing between transmission of safe values and safe code.



< □ > < □ > < □ > < □ > < □ >

- A type theory including modal operators with judgmental scope;
 - "command execution for A is valid at every address" □(A true);
 - "command execution for A is valid at a given address" \Diamond (A true).



- A type theory including modal operators with judgmental scope;
 - "command execution for A is valid at every address" □(A true);
 - "command execution for A is valid at a given address" \diamond (A true).
- Modalities generated by polymorphism of constructions to express overall executability or broadcasting of location-bounded code;



- A type theory including modal operators with judgmental scope;
 - "command execution for A is valid at every address" $\Box(A true)$;
 - "command execution for A is valid at a given address" \diamond (A true).
- Modalities generated by polymorphism of constructions to express overall executability or broadcasting of location-bounded code;
- Indexed multimodalities to localize data and express interaction of commands;



- A type theory including modal operators with judgmental scope;
 - "command execution for A is valid at every address" □(A true);
 - "command execution for A is valid at a given address" \diamond (A true).
- Modalities generated by polymorphism of constructions to express overall executability or broadcasting of location-bounded code;
- Indexed multimodalities to localize data and express interaction of commands;
- Contexts describe resources of networks in which code is executed; order of assumptions is used to mimick the composition of commands.



A modal contextual type theory with judgemental modalities







G. Primiero (Ghent University)

Image: A matrix

- N

Structure of the language

- polymorphic language: \mathcal{K} : {*type*, *type*_{inf}}
 - type: computations with complete instructional informations;
 - type_{inf}: computational instructions admissible to execute a command.



Structure of the language

• polymorphic language: \mathcal{K} : {*type*, *type*_{inf}}

- type: computations with complete instructional informations;
- type_{inf}: computational instructions admissible to execute a command.
- Туре
 - type-constructors composed by listing, application, abstraction and pairing for ∧, ∨, →, ∀, ∃;
 - \rightarrow as a λ -term presented *together with* one of its α -terms;
 - a_i : A induces $\Box_i(A true)$:
 - computations that can be safely run everywhere;
 - $\Box_i(A \text{ true}) \text{ induces } \Box_j(A \text{ true}) \text{ for all } i, j;$



Structure of the language

• polymorphic language: \mathcal{K} : {*type*, *type*_{inf}}

- type: computations with complete instructional informations;
- type_{inf}: computational instructions admissible to execute a command.
- Туре
 - type-constructors composed by listing, application, abstraction and pairing for ∧, ∨, →, ∀, ∃;
 - \rightarrow as a λ -term presented *together with* one of its α -terms;
 - a_i : A induces $\Box_i(A true)$:
 - computations that can be safely run everywhere;
 - $\Box_i(A \text{ true})$ induces $\Box_j(A \text{ true})$ for all i, j;
- Type_{inf}
 - Admissibility defined from $\neg(A \rightarrow \bot)$ to x : A;
 - ► ⊃ is composition by abstraction (admissible command at address);
 - x_i : A induces \diamond_i (A true):



- address-bounded computations;
- the given location needs to be called upon to produce safely a value;

Definition (The set of terms)

The set \mathcal{T} of terms is given by:

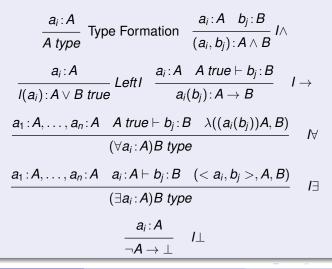
 $\mathcal{T} := \left\{ \begin{array}{l} a_i; (a_i, b_j); a_i(b_j); \lambda(a_i(b_j)); < a_i, b_j >; \text{ (Constructors for terms)}; \\ x_i; (x_i(b_j)); (x_i(b_j))(a_i), \text{ (Variables for terms)}. \end{array} \right.$



Interpreting complete code (1)

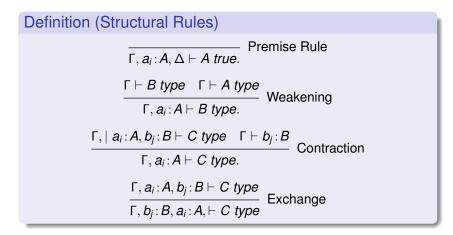
Definition (Rules for type)

The rules for signed expressions in the kind type are:



G. Primiero (Ghent University)

Interpreting complete code (2)





Interpreting executable code (1)

Definition (Rules for type_{inf})

The rules for signed expressions in the kind type_{inf} are:

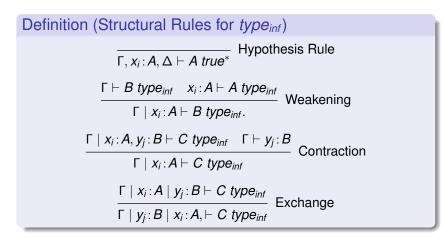
$$\frac{\neg (A \to \bot) \ type \quad x_i : A}{A \ type_{inf}} \ Type_{inf} \ Formation$$

$$\frac{A \ type_{inf} \quad b_j : B[x_i : A]}{((x_i)b_j) : A \supset B \ true} \ Functional \ abstraction$$

$$\frac{A \ type_{inf} \quad b_j : B[x_i : A] \quad a_i : A}{(x(b_j))(a_i) = b[a/x] : B \ type[a/x]} \ \beta - conversion$$

$$\frac{\lambda((a_{1-i}(b_j))A, B) \quad (b_j)[a_i := a]}{(a_i(b_j)) : A \to B} \ \alpha - conversion$$

Interpreting executable code (2)





ヘロト 人間ト 人間ト 人用ト

Language (2)

Definition (Modal Judgements)

The set of modal judgements M for any $i \in G$ is defined by the following modal formation rules:

$$\frac{a_i:A}{\Box_i(A \ true)} \Box - \text{Formation} \quad \frac{x_i:A}{\diamond_i(A \ true)} \diamond - \text{Formation}$$



Generalized Contextual Format

Definition (Signed and Modal Contexts)

- Γ_i is a context signed for *i* iff Γ_i = {x_i: A,..., x_i: N}, all with distinct subjects {A,..., N} ∈ type_{inf};
- ② □_{*i*}(*A true*) is a modal premise generated from $[x_i/a_i]$: *A* with $x_i, a_i \in \mathcal{T}$, and *A type*;
- Solution (A true) is a modal assumption generated from x_i: A with x_i ∈ T and A type_{inf};
- So For any context Γ_i , $\Box_i \Gamma$ is given by $\bigcup \{ \Box_i (A \text{ true}) \mid \text{ for all } A \in \Gamma \};$
- For any context Γ_i, \diamond_i Γ is given by \bigcup { \circ_i (*A true*) | \circ = {□, \diamond } and \diamond_i (*A true*) for at least one *A* ∈ Γ}.



< ロ > < 同 > < 回 > < 回 > < 回 > < 回

Generalized Contextual Format

Definition (Signed and Modal Contexts)

- Γ_i is a context signed for *i* iff $\Gamma_i = \{x_i : A, \dots, x_i : N\}$, all with distinct subjects $\{A, \dots, N\} \in type_{inf}$;
- ② □_{*i*}(*A true*) is a modal premise generated from $[x_i/a_i]$: *A* with $x_i, a_i \in T$, and *A type*;
- Solution (A true) is a modal assumption generated from x_i: A with x_i ∈ T and A type_{inf};
- So For any context Γ_i , $\Box_i \Gamma$ is given by $\bigcup \{ \Box_i (A \text{ true}) \mid \text{ for all } A \in \Gamma \};$
- So For any context Γ_i , $\diamond_i \Gamma$ is given by $\bigcup \{ \circ_i (A \text{ true}) \mid \circ = \{ \Box, \diamond \} \}$ and $\diamond_i (A \text{ true})$ for at least one $A \in \Gamma \}$.
 - Terms in □_iΓ refer to commands that can be broadcasted from original address *i* to any other address in the Network for execution;



Terms in $\diamond_i \Gamma$ refer to commands that need to be executed explicitly at the originating address *i*.

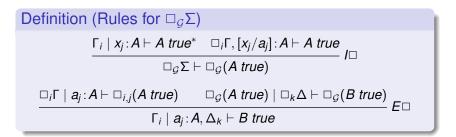
Derivability from Modal Contexts

Definition (Modal Judgements from multi-signed contexts)

- $\Box_k(A \text{ true})$ iff for all $\Gamma_j \in Context$, $\emptyset \mid \Box_j \Gamma \vdash \Box_k(A \text{ true})$, where $j = \bigcup \{1, \ldots, k-1\} \in \mathcal{G};$
- $\diamond_k(A \text{ true})$ iff for some $\Gamma_i, \Delta_j \in Context$, $\Box_i \Gamma \mid \diamond_j \Delta \vdash \diamond_k(A \text{ true})$, where $j = \bigcup \{1, \dots, k-1\} \in \mathcal{G};$



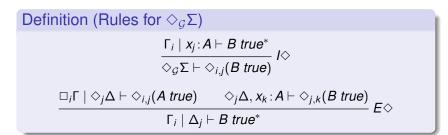
Introduction and Elimination for \Box



- If executing A requires code at *j* which can be broadcasted elsewhere within network Σ, then program A can be executed everywhere in Σ.
- Elimination sends expression (*A true*) from *i*, *j* to *k* where *B* can be evaluated.



Introduction and Elimination for \diamond



- If executing B requires code bounded at i, j, then those adresses are required within network Σ;
- Introduction constructs a return value for *B* executed at *i*, *j*; Elimination is the semantic counterpart of extracting code at sources.



Code Mobility Rules

Definition (Broadcast)

Broadcacst is used to send to a specific address an exec command that can be executed everywhere in the network Σ .

$$\frac{\Box_i \Gamma, a_j : A \vdash \Box_{i,j}(B \text{ true}) \quad x_j : A \vdash A \text{ true}^*}{\Box_i \Gamma, \diamond_j(A \text{ true}) \vdash \diamond_{i \cap j}(B \text{ true})}$$

(Broadcast)



Code Mobility Rules

Definition (Broadcast)

Broadcacst is used to send to a specific address an exec command that can be executed everywhere in the network Σ .

$$\frac{\Box_i \Gamma, a_j : A \vdash \Box_{i,j} (B true)}{\Box_i \Gamma, \diamond_j (A true) \vdash \diamond_{i \cap j} (B true)} x_j : A \vdash A true^*$$

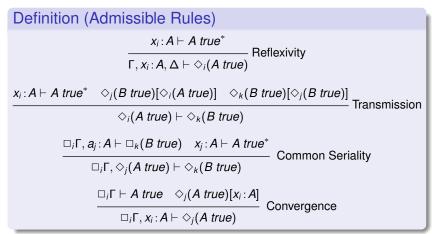
(Broadcast)

Definition (Global Access)

Global Access is the reverse function that calls from a specific address within network Σ a command that becomes executable at any address.

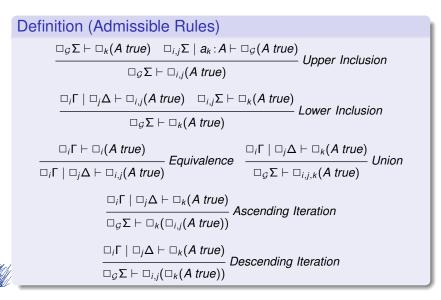
$$\frac{ \Gamma_i, x_j : A \vdash B \ true^* \quad a_j : A \vdash A \ true}{\Box_i \Gamma, a_j : A \vdash \Box_{i \cup j} (B \ true)} \quad (Global \ Access)$$

Properties





Properties (cnt'd)



< □ > < □ > < □ > < □ > < □ > < □ >

Properties (cnt'd)

- Local soundness and completeness generalized from the expansion/reduction in the monomodal case from [Primiero(2010)];
- Substitution on terms and truth predicates:
 - If $\Gamma_i, x_j : A, \Delta_k \vdash B$ true^{*} and $\Gamma_i, \Delta_k \vdash a_j : A$, then $\Gamma_i, \Delta_k \vdash [x/a]_{i,k}B$ true.
 - ② If □_iΓ, \diamond_j (*A* true), □_kΔ ⊢ $\diamond_{\cap(i,j,k)}$ (*B* true) and □_iΓ, □_jΔ ⊢ □_{∪(i,j)}(*A* true), then □_iΓ, □_jΔ ⊢ □_{∪(i,j)}(*B* true).





2 A modal contextual type theory with judgemental modalities







Syntax

Definition (Syntax of the Programming Language)

Types := $\alpha; \tau_1 \land \tau_2; \tau_1 \lor \tau_2; \tau_1 \rightarrow \tau_2; \tau_1 \supset \tau_2; \neg \tau \rightarrow \bot; \Box \tau, \Diamond \tau$ Terms := $x_i \mid a_i \quad constants, variables$ $(a_i, b_j) \quad pairs$ $exec(\alpha) \mid run_i(\alpha) \mid synchro_j(run_i(\alpha)) \quad functions$ $GLOB(\Box \Gamma, \alpha_{i\cup j}) \mid BROAD(\Diamond \Gamma, \alpha_{i\cap j}) \quad remote$ $RET(\Gamma, \alpha_{i\cup j}) \mid SEN(\Gamma, \alpha_{i\cap j}) \quad portable$ Contexts := $\Gamma_i, \Box_i \Gamma; \Diamond_i \Gamma$.



< □ > < □ > < □ > < □ > < □ > < □ >

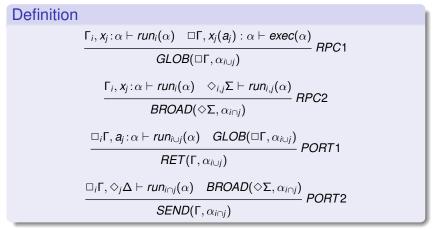
Syntax (2)

Definition (Typing Rules)

$$\frac{\overline{\Gamma, a_{i}: \alpha, \Delta \vdash exec(\alpha).}}{run_{i\cup j}(\alpha \times \beta)} \begin{array}{c} \text{global} & \overline{\Gamma, x_{i}: \alpha, \Delta \vdash run_{i}(\alpha).} \end{array} \begin{array}{c} \text{local} \\ \\ \hline \hline{\Gamma, a_{i}: \alpha, \Delta \vdash exec(\alpha).} \end{array} \begin{array}{c} I \land & \frac{(a_{i}, b_{j}): \alpha \times \beta}{exec(\alpha)} E \land (1) \\ \hline \hline{a_{i}: \alpha \quad exec(\alpha) \vdash b_{j}: \beta} \\ \hline run_{i\cup j}(\alpha \to \beta) \end{array} \begin{array}{c} I \rightarrow & \frac{x_{i}: \alpha \quad run_{i}(\alpha) \vdash b_{j}: \beta}{run_{i\cap j}(\alpha \supset \beta)} \end{array} \begin{array}{c} I \supset \\ \hline \hline \\ \hline \hline x_{i}: \alpha \vdash run_{i}(\alpha) & x_{i}(b_{j}): \alpha \supset \beta \\ \hline \end{array} \begin{array}{c} x_{i}: \alpha \vdash run_{i}(\alpha) & x_{i}(b_{j}): \alpha \supset \beta \\ \hline \end{array} \begin{array}{c} x_{j}: \alpha \vdash run_{i}(\alpha) & x_{i}(b_{j}): \alpha \supset \beta \\ \hline \end{array} \end{array}$$



Syntax (3)





< □ > < □ > < □ > < □ > < □ > < □ >

Operational Semantics

Definition (Operational Model)

Networks $\mathcal{N} := (\mathcal{I}, \mathcal{L})$ Process Environments $\mathcal{L} := (I.i \mapsto e) \mid i \in \mathcal{I}, i \in \mathcal{T}$ Terms $\mathcal{T} := | synchro(.(.)) \mid run_i(\alpha) \mid exec(\alpha) \mid GLOB(.) \mid BROAD(.) \mid$ Contexts $\mathcal{C} := \Gamma \mid \circ \Gamma \mid (\Gamma, e) \mid$



.

Operational Semantics (II)

	$(L)\mapsto (L')$
run	$[\Gamma]x_i \mapsto [\Diamond \Gamma]run_i(\alpha)$
exec	$[\Gamma]a_i \mapsto [\Box \Gamma]exec(\alpha)$
\rightarrow	$[\Gamma]exec(\alpha) \vdash b_j \mapsto [\Box \Gamma]synchro(b_j(exec(\alpha)))$
\supset	$[\Gamma]$ run _i $(\alpha) \vdash b_j \mapsto [\Diamond \Gamma]$ synchro $(b_j(run_i(\alpha)))$
\wedge	$[\Gamma] run_i(\alpha), run_j(\beta) \mapsto [\Box \Gamma] exec(\alpha, \beta)$
□1	$[\Gamma] \vdash run_i(\alpha) \mapsto GLOB[\Box \Gamma, \alpha]$
□2	$[\Box_i \Gamma] exec(\alpha) \mapsto RET[\Gamma, \alpha_{i \cup j}]$
◇1	$[\Gamma] \vdash run_i(\alpha) \mapsto BROAD[\Diamond \Gamma, \alpha_{i \cap j}]$
♦2	$[\Diamond \Gamma] \vdash run_{i,j}(\alpha) \mapsto SEND[\Gamma, \alpha_{i \cap j}]$



∃ >

Evaluation

- Evaluation of syntactic transformation requires all closed expressions (*exec*(α) and [□Γ]α);
- Occurrences of *run_i* and *◊_i* require a side condition on preservation of indices;
- Evaluation on contexts proceeds on the ordering induced by *i* < *j*;
- A transition $\mathcal{L}\mapsto \mathcal{L}'$ consists of
 - decomposing L into an evaluation context (if present) and an instruction;
 - evaluation of the context and execution of the instruction;
 - replacement of intruction execution in one of the rules to obtain L'.



Safety

Theorem (Type Safety)

• If
$$e : \alpha$$
 for $\mathcal{L} := (I.i \mapsto e)$, and $\mathcal{L} \mapsto \mathcal{L}'$, then $e' : \alpha$ for $\mathcal{L}' := (I.i \mapsto e')$;

If e: α for L := (I.i → e), then either exec(α) is the output value or there is e' for L' := (I.i → e') s.t. L → L'.

Theorem (Preservation)

If $[\Gamma]e: \alpha$ for $\mathcal{L} := (I.i \mapsto e)$, and $\mathcal{L} \mapsto \mathcal{L}'$, then there is $[\Box \Gamma]e': \alpha$ for $\mathcal{L}' := (I.i \mapsto e')$

Theorem (Progress)

If $[\Box \Gamma]e: \alpha$ for $\mathcal{L} := (l.i \mapsto e)$, then either $\mathcal{L} \mapsto \mathcal{L}'$ or $exec(\alpha)$ is the output value.

.



2 A modal contextual type theory with judgemental modalities





Multi-Modal Type Theory

PSPL2010 28 / 31

Conclusions

- A Computational Interpretation for a Multimodal Type-Theory with indexed and ordered Contexts;
- Corresponding Epistemic Interpretation for Trusted Communications;
- Models:
 - Weakening of the poset {1,0} that satisfies inhabitness and intensional identity;
 - * $A = [a, \rightarrow] = \{1\}$ if $x \rightarrow a = 1$ and A: type = 1
 - * $A = [a, \rightarrow] = \emptyset$ if $x \rightarrow a =$ undefined and $A: type_{inf} = 1$
 - * $A = [a, \rightarrow] = \{0\}$ if $x \rightarrow a = 0$ and A: type = 0
 - type_{inf} admits undefinability, preserves only symmetricity; inhabitness is not guaranteed ('super-modest types');
 - Semantics of *cKT*_{□,◊} obtained by a composed set of (non-standard) Kripke models *M*^(L^{ver}∪L^{inf}).



References

N. Alechina, M. Mendler, V. de Paiva, and E. Ritter. Categorical and Kripke Semantics for Constructive S4 Modal Logic.

In Proceedings of the 15th International Workshop on Computer Science Logic, volume 2142 of Lecture Notes In Computer Science, pages 292 – 307, 2001.

- G.M. Bierman and V. de Paiva. On an intuitionistic modal logic. *Studia Logica*, (65):383–416, 2000.

IIIII

T. Borghuis and L.M.G. Feijs.

A constructive logic for services and information flow in computer networks.

The Computer Journal, pp.274–289, vol.43, n.4, 2000.

- R. Davies and F. Pfenning.
- A modal analysis of staged computation.

[™] Journal of the ACM, 48(3):555–604, 2001.

References

L. Jia and D. Walker.

Modal Proofs as Distributed Programs.

In *Programming Languages and Systems, ESOP2004*, volume 2986 of *Lectures Notes in Computer Science*. Springer Verlag, 2004.

J. Moody.

Modal logic as a basis for distributed computation.

Technical Report CMU-CS-03-194, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, USA, 2003.

T. Murphy.

Modal Types for Mobile Code.

PhD thesis, School of Computer Science, Carnegie Mellon University, 2008. CMU-CS-08-126.



T. Murphy, K. Crary, and R. Harper. *Type-Safe Distributed Programming with ML5*, volume 4912 of *Lectures Notes in Computer Science*, pages 108–123. Springer Verlag, 2008.

References

- A. Nanevski, F. Pfenning, and B. Pientka. Contextual modal type theory. ACM Transactions on Computational Logic, 9(3):1–48, 2008.
- F. Pfenning and R. Davies.

A judgemental reconstruction of modal logic.

Mathematical Structures in Computer Science, 11:511–540, 2001.

S. Park.

A modal language for the safety of mobile values. In Fourth ASIAN Symposium on Programming Languages and Systems, 2006, pp.217–233, Springer.

G. Primiero.

Constructive contextual modal judgments for reasoning from open assumptions.

In Proceedings of the Computability in Europe Conference, 2010.



The Proof Theory and Semantics of Intuitionistic Modal Logic.

PhD thesis, University of Edinburgh, College of Science and

G. Primiero (Ghent University)

Multi-Modal Type Theory

PSPL2010 31/31