Multi-level Lax Logic

Edwin Lewis-Kelham   Mike Stannett

Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK.
Correspondence: `M.Stannett@dcs.shef.ac.uk`

Many thanks to the EPSRC for funding Ed's research.

TYPES 2006, 18 April 2006

Outline

Lax Logic [Men93, FM97]

- Given a base logic $\mathcal{B}$
- we can define a first-order logic, $\mathcal{L}$, equipped with
- a modality, $\bigcirc$, and
- a unary connective ι that faithfully embeds propositions of $\mathcal{B}$ as formulae of $\mathcal{L}$.

The modality represents the idea that a statement can be validated relative to some — initially unspecified — constraint. The statement $\bigcirc\phi$ ('somehow ϕ') is intended to mean 'for some constraint c, ϕ holds under c'.

History: Recent

- originally developed by Mendler [Men93] for extracting and reasoning about constraints during hardware verification and refinement.
- propositional lax logic (PLL) developed by Mendler and Fairtlough [FM97]
- two quantified versions (QLL, QLL^+) developed by Fairtlough and Walton [FW97, Wal99]
- multi-level version (MLL) developed by Ed Lewis as part of his PhD work [LK06] — described below

History: Ancient

With hindsight, $\bigcirc$ has been studied in other contexts for æons.

- Earliest reference(?) is Curry's presentation of *an elimination theorem in the presence of modality* [Cur52]
- Aczel [Acz99] has identified lax modalities occurring as
  - *nuclei* in locale theory
  - *strong monads* on categories
  - *modalities* in topos theory.
- Pfenning and Davies [PD99] showed lax logic is contained within modal logic via $\bigcirc P \equiv \Diamond\Box P$ with $P \rightarrow_{\mathcal{L}} Q \equiv (\Box P) \rightarrow Q$.

Base Logic

$\mathcal{B}$ should be many-sorted logic, with equality $=$, implication $\rightarrow$, quantification $\forall$, sorts S (including propositions, Ω) and operators O.

Types

$$\tau ::= A \mid \mathbf{0} \mid \mathbf{1} \mid \tau + \tau \mid \tau \times \tau \mid \tau \Rightarrow \tau \mid \tau^{*} \mid \mathbb{N}$$

where $A \in S$. Quantification is allowed over any type, e.g. $\neg\phi =_{\mathrm{def}} \phi \rightarrow \mathit{false}$ where $\mathit{false} =_{\mathrm{def}} \forall x^{\Omega}.x$.

Terms

$$\begin{aligned} t ::= \quad & x \mid f(t, \ldots, t) \mid t \rightarrow t \mid \forall x.t \mid t = t \mid \\ & * \mid \pi_L\, t \mid \pi_R\, t \mid (t, t) \mid \\ & t\ t \mid \lambda x.t \mid \Box t \mid \mathrm{in}_L\, t \mid \mathrm{in}_R\, t \mid \mathit{case}_{x,y}(t, t, t) \mid \\ & [\,] \mid t :: t \mid \mathit{fold}_{x,z}(t, t) \mid 0 \mid \mathit{succ} \mid \mathit{iter}_x(t, t) \end{aligned}$$

where x, y, z are variables and $f \in O$.

Base: Induction principles and equality axioms

$$\frac{\Gamma \vdash_{\mathcal{B}_\Delta} \phi\{[\,]/z\} \quad \Gamma, \phi \vdash_{\mathcal{B}_{\Delta,x,z}} \phi\{x :: z/z\}}{\Gamma \vdash_{\mathcal{B}_\Delta} \forall z.\phi}\ ListInd \qquad \frac{\Gamma \vdash_{\mathcal{B}_\Delta} \phi\{0/z\} \quad \Gamma, \phi \vdash_{\mathcal{B}_{\Delta,z}} \phi\{succ\ z/z\}}{\Gamma \vdash_{\mathcal{B}_\Delta} \forall z.\phi}\ N$$

$$\frac{\Delta, x^{\mathbf{0}} \vdash_{\mathcal{B}} t : \tau}{\vdash_{\mathcal{B}_{\Delta,x^{\mathbf{0}}}} t = \Box x} \qquad \frac{\Delta, x^{\sigma} \vdash_{\mathcal{B}} t : \tau}{\vdash_{\mathcal{B}_{\Delta,x^\sigma}} (\lambda x.t)x = t}$$

$$\frac{}{\vdash_{\mathcal{B}_{x^\sigma,y^\tau}} \pi_L(x,y) = x} \qquad \frac{}{\vdash_{\mathcal{B}_{x^\sigma,y^\tau}} \pi_R(x,y) = y}$$

$$\frac{\Delta \vdash_{\mathcal{B}} u : \sigma_1 \quad \Delta, x^{\sigma_1} \vdash_{\mathcal{B}} s : \tau \quad \Delta, y^{\sigma_2} \vdash_{\mathcal{B}} t : \tau}{\vdash_{\mathcal{B}_\Delta} case_{x,y}(\mathrm{in}_L\ u, s, t) = s\{u/x\}} \qquad \frac{\Delta \vdash_{\mathcal{B}} u : \sigma_1 \quad \Delta, x^{\sigma_1} \vdash_{\mathcal{B}} s : \tau \quad \Delta, y^{\sigma_2} \vdash_{\mathcal{B}} t}{\vdash_{\mathcal{B}_\Delta} case_{x,y}(\mathrm{in}_R\ u, s, t) = s\{u/y\}}$$

$$\frac{\Delta \vdash_{\mathcal{B}} s : \tau \quad \Delta, x^{\tau} \vdash_{\mathcal{B}} t : \tau}{\vdash_{\mathcal{B}_\Delta} iter_x(s,t)0 = s} \qquad \frac{\Delta \vdash_{\mathcal{B}} s : \tau \quad \Delta, x^{\tau} \vdash_{\mathcal{B}} t : \tau}{\vdash_{\mathcal{B}_{\Delta,z^{\mathbb{N}}}} iter_x(s,t)(succ\ z) = t\{iter_x(s,t)z/x\}}\ z^{\mathbb{N}} \notin \Delta$$

$$\frac{\Delta \vdash_{\mathcal{B}} s : \sigma \quad \Delta, z^{\tau}, x^{\sigma} \vdash_{\mathcal{B}} t : \sigma}{\vdash_{\mathcal{B}_\Delta} fold_{z,x}(s,t)[\,] = s} \qquad \frac{\Delta \vdash_{\mathcal{B}} s : \sigma \quad \Delta, z^{\tau}, x^{\sigma} \vdash_{\mathcal{B}} t : \sigma}{\vdash_{\mathcal{B}_\Delta} fold_{z,x}(s,t)(u :: v) = t\{(fold_{z,x}(s,t)v)/x\}\{u/z\}}\ v^{\tau^*}, u^{\tau}$$

$$\frac{}{\vdash_{\mathcal{B}_{x^{\mathbf{1}}}} x = *} \qquad \frac{}{\vdash_{\mathcal{B}_{x^{\sigma\times\tau}}} (\pi_L\ x, \pi_R\ x) = x} \qquad \frac{\Delta \vdash_{\mathcal{B}} t : \sigma \Rightarrow \tau}{\vdash_{\mathcal{B}_\Delta} \lambda x.(tx) = t}\ x^{\sigma} \notin \Delta$$

$$\frac{\Delta, z^{\sigma_1+\sigma_2} \vdash_{\mathcal{B}} h : \tau}{\vdash_{\mathcal{B}_{\Delta,z^{\sigma_1+\sigma_2}}} case_{x,y}(z, h\{\mathrm{in}_L\ x/z\}, h\{\mathrm{in}_R\ y/z\} = h)}\ x^{\sigma_1}, y^{\sigma_2} \notin \Delta$$

$$\frac{\vdash_{\mathcal{B}_{\Delta,x}} s = t}{\vdash_{\mathcal{B}_\Delta} \lambda x.s = \lambda x.t} \qquad \frac{\Delta \vdash_{\mathcal{B}} u : \sigma_1 + \sigma_2 \quad \vdash_{\mathcal{B}_{\Delta,x^{\sigma_1}}} s = s' \quad \vdash_{\mathcal{B}_{\Delta,y^{\sigma_2}}} t = t'}{\vdash_{\mathcal{B}_\Delta} case_{x,y}(u, s, t) = case_{x,y}(u, s', t')}$$

Lax: Formulae

The formulae M of $\mathcal{L}$ are given by

$$\begin{aligned} M ::= \quad & \iota\phi \mid true \mid false \mid \bigcirc M \mid \\ & M \wedge M \mid M \vee M \mid M \rightarrow M \mid \forall x.M \mid \exists x.M \end{aligned}$$

where ϕ ranges over the propositions of $\mathcal{B}$ and x ranges over variables. The role of each connective (i.e. whether it is in $\mathcal{B}$ or $\mathcal{L}$) is always clear from context.

Lax: Deduction Rules

Most of these rules are standard.

$$\frac{}{true}\;trueI \qquad \frac{false}{M}\;falseE \qquad \frac{M \quad N}{M \wedge N}\;\wedge I \qquad \frac{M \wedge N}{M}\;\wedge E_L \qquad \frac{M \wedge N}{N}\;\wedge E_R$$

$$\frac{M}{M \vee N}\;\vee I_L \qquad \frac{N}{M \vee N}\;\vee I_R \qquad \frac{M_1 \vee M_2 \quad \begin{array}{c}[x_1 : M_1]\\ \vdots \\ N\end{array} \quad \begin{array}{c}[x_2 : M_2]\\ \vdots \\ N\end{array}}{N}\;\vee E_{x_1,x_2}$$

$$\frac{M}{\forall x.M}\;\forall I_x \qquad \frac{\forall x.M}{M\{t/x\}}\;\forall E_t \qquad \frac{\exists x.M \quad \begin{array}{c}[y : M]\\ \vdots \\ N\end{array}}{N}\;\exists E_y \qquad \frac{M\{t/x\}}{\exists x.M}\;\exists I_t$$

$$\frac{\iota\phi_1 \ldots \iota\phi_k}{\iota\psi}\;\iota \textbf{ (side condition: } \phi_1, \ldots, \phi_k \vdash_{\mathcal{B}} \psi\textbf{)}$$

$$\frac{\iota(s = t) \quad M\{s/x\}}{M\{t/x\}}\;Subst \qquad \frac{\begin{array}{c}[x : M]\\ \vdots \\ N\end{array}}{M \rightarrow N}\;\rightarrow I_x \qquad \frac{M \rightarrow N \quad M}{N}\;\rightarrow E$$

Lax: Deduction rules (cont.)

Mendler's `lets-not-bother` rule is a bit odd! Even though it provides no information, it still seems to be useful (worth investigating further).

$$
\dfrac{\bigcirc M \qquad \begin{array}{c}[x : M]\\ \vdots \\ \bigcirc N\end{array}}{\bigcirc N}\ \bigcirc L_x
\qquad
\dfrac{M}{\bigcirc M}\ \bigcirc I
\qquad
\dfrac{\bigcirc\bigcirc M}{\bigcirc M}\ \bigcirc M
\qquad
\dfrac{\bigcirc M \qquad \begin{array}{c}[x : M]\\ \vdots \\ N\end{array}}{\bigcirc N}\ \bigcirc F_x
$$

$$
\dfrac{M\{0/n\} \qquad \begin{array}{c}[x : M]\\ \vdots \\ M\{succ\ n/n\}\end{array}}{\forall n.M}\ NatInd_{n,x}
\qquad
\dfrac{M\{[\,]/l\} \qquad \begin{array}{c}[x : M]\\ \vdots \\ M\{h :: l/l\}\end{array}}{\forall l.M}\ ListInd_{h,l,x}
$$

$$
\dfrac{}{\bigcirc M}\ \texttt{lets-not-bother}
$$

Lax: Constraint extraction

A proof of $\bigcirc\phi$ is a pair (c, p) where c is a constraint and p is a proof of ϕ under c. We need to find both c and p. We first associate every closed $\mathcal{L}$-statement M with a predicator $M^{\#}$.

$$
\boxed{
\begin{array}{rcl}
(\iota\phi)^{\#} z & =_{\mathrm{def}} & \phi \\
(\bigcirc M)^{\#} z & =_{\mathrm{def}} & (M^{\#}(\pi_R\, z))^{\pi_L\, z} \\
false^{\#} z & =_{\mathrm{def}} & false \\
true^{\#} z & =_{\mathrm{def}} & true \\
(M \wedge N)^{\#} z & =_{\mathrm{def}} & M^{\#}(\pi_L\, z) \wedge N^{\#}(\pi_R\, z) \\
(M \vee N)^{\#} z & =_{\mathrm{def}} & (\exists x^{|M|}.z = \mathrm{in}_L\, x \wedge M^{\#} x) \vee \\
 & & (\exists y^{|N|}.z = \mathrm{in}_R\, y \wedge N^{\#} y) \\
(M \rightarrow N)^{\#} z & =_{\mathrm{def}} & \forall x^{|M|}.M^{\#} x \rightarrow N^{\#}(zx) \\
(\forall x^{\tau}.M)^{\#} z & =_{\mathrm{def}} & \forall x^{\tau}.M^{\#}(zx) \\
(\exists x^{\tau}.M)^{\#} z & =_{\mathrm{def}} & (M\{\pi_L\, z/x\})^{\#}(\pi_R\, z)
\end{array}
}
$$

Lax: Constraint extraction (cont.)

Next we find any proof of $\bigcirc\phi$ and translate it using these rules:

$$
\begin{array}{rcl}
[trueI] & = & * \\
[falseE(a)] & = & \Box[a] \\
[\wedge I(a,b)] & = & ([a],[b]) \\
[\wedge E_L(a)] & = & \pi_L[a] \\
[\wedge E_R(a)] & = & \pi_R[a] \\
[\vee I_L(a)] & = & \mathrm{in}_L[a] \\
[\vee I_R(a)] & = & \mathrm{in}_R[a] \\
[\vee E_{x_1,x_2}(a,b_1,b_2)] & = & case_{x_1,x_2}([a],[b_1],[b_2]) \\
[\forall I_x(a)] & = & \lambda x.[a] \\
[\forall E_t(a)] & = & [a]\ t \\
[\exists E_y(a,b)] & = & [b]\{\pi_L[a]/x\}\{\pi_R[a]/y\} \\
[\exists I_t(a)] & = & (t,[a]) \\
[\iota(a_1,\ldots,a_k)] & = & * \\
[\rightarrow I_x(a)] & = & \lambda x.[a] \\
[\rightarrow E(a,b)] & = & [a]\ [b] \\
[\bigcirc L_x(a,b)] & = & (\pi_L([b]\{\pi_R[a]/x\})@\,\pi_L[a],\pi_R([b]\{\pi_R[a]/x\})) \\
[\bigcirc I(a)] & = & ([\,],[a]) \\
[\bigcirc M(a)] & = & ((\pi_L\ \pi_R[a])@(\pi_L[a]),\pi_R\ \pi_R[a]) \\
[\bigcirc F_x(a,b)] & = & (\pi_L[a],[b]\{\pi_R[a]/x\}) \\
[Subst(a,b)] & = & [b] \\
[NatInd_{n,x}(a,b)] & = & natrec([a],\lambda n.\lambda x.[b]) \\
[ListInd_{h.l,x}(a,b)] & = & listrec([a],\lambda h.\lambda l.\lambda x.[b])
\end{array}
$$

Example

Consider the formula

$$SPEC =_{\text{def}} \forall m^{\mathbb{N}}.\bigcirc\iota\exists n^{\mathbb{N}}.(m = succ\ n) \quad .$$

Example

Consider the formula

$$SPEC =_{\mathrm{def}} \forall m^{\mathbb{N}}.\bigcirc\iota\exists n^{\mathbb{N}}.(m = succ\ n) \ \ .$$

We expect to extract '$m \neq 0$'.

Example

Consider the formula

$$SPEC =_{\mathrm{def}} \forall m^{\mathbb{N}}.\bigcirc\iota\exists n^{\mathbb{N}}.(m = succ\ n) \ .$$

We expect to extract '$m \neq 0$'. Given any constraint term z, we get

$$\begin{aligned}
SPEC^{\#} z =&(\forall m^{\mathbb{N}}.\bigcirc\iota\exists n^{\mathbb{N}}.(m = succ\ n))^{\#} z \\
=&\forall m^{\mathbb{N}}.((\bigcirc\iota\exists n^{\mathbb{N}}.(m = succ\ n))^{\#}(zm)) \\
=&\forall m^{\mathbb{N}}.(((\iota\exists n^{\mathbb{N}}.(m = succ\ n))^{\#}\, \pi_R(zm))^{\pi_L(zm)}) \\
=&\forall m^{\mathbb{N}}.((\exists n^{\mathbb{N}}.(m = succ\ n))^{\pi_L(zm)}) \\
=&\forall m^{\mathbb{N}}.(\phi^{\pi_L(zm)})
\end{aligned}$$

where $\phi =_{\mathrm{def}} \exists n^{\mathbb{N}}.(m = succ\ n)$, so the constraint in question is given by the subterm $\pi_L(zm)$.

Example (cont).

Different proofs of $SPEC$ yield different choices for z. Let's use the following proof.

$$
\dfrac{
\dfrac{w}{\bigcirc\iota\exists n^{\mathbb{N}}.(0 = succ\ n)}\ \forall E_{\exists n^{\mathbb{N}}.(0 = succ\ n)}
\qquad
\dfrac{\dfrac{\dfrac{\begin{array}{c}[m : \mathbb{N}] \\ \vdots \\ succ\ m = succ\ m\end{array}}{\exists n^{\mathbb{N}}.(succ\ m = succ\ n)}\ \exists_{\mathcal{B}} I}{\iota\exists n^{\mathbb{N}}.(succ\ m = succ\ n)}\ \iota}{\bigcirc\iota\exists n^{\mathbb{N}}.(succ\ m = succ\ n)}\ \bigcirc I
}{\forall m^{\mathbb{N}}.\bigcirc\iota\exists n^{\mathbb{N}}.(m = succ\ n)}\ NatInd_{m,m}
$$

Example (cont.)

This translates into the constraint term

$$
\begin{aligned}
z &= [NatInd_{m,m}]([\forall E_{\exists n^{\mathbb{N}}.(0 = succ\ n)}]([w]), [\bigcirc I]([\iota]([\ldots])))\\
&= natrec\ ([\forall E_{\exists n^{\mathbb{N}}.(0 = succ\ n)}](?), \lambda m.\lambda m'.[\bigcirc I]([\iota]([\ldots])))\\
&= natrec\ (?(\exists n^{\mathbb{N}}.(0 = succ\ n)), \lambda m.\lambda m'.([\,], [\iota]([\ldots])))\\
&= natrec\ (([\exists n^{\mathbb{N}}.(0 = succ\ n)], *), \lambda m.\lambda m'.([\,], *))
\end{aligned}
$$

and the required constraint, $\pi_L(zm)$, is

$$\pi_L(zm) \equiv \pi_L(natrec(([\exists n^{\mathbb{N}}.(0 = succ\ n)], *), \lambda m.\lambda m'.([\,], *))\ m)$$

Example (cont.)

This is equivalent to $(m \neq 0)$, as required: For the base case $(m = 0)$, we have

$$\begin{aligned}
\pi_L(z\ 0) &\equiv \pi_L(natrec(([\exists n^{\mathbb{N}}.(0 = succ\ n)], *), \lambda m.\lambda m'.([\,], *))\ 0) \\
&\equiv \pi_L([\exists n^{\mathbb{N}}.(0 = succ\ n)], *) \\
&\equiv [\exists n^{\mathbb{N}}.(0 = succ\ n)]
\end{aligned}$$

and for $m = succ\ k$ we have

$$\begin{aligned}
\pi_L(z\ (succ\ k)) &\equiv \pi_L(natrec(([\exists n^{\mathbb{N}}.(0 = succ\ n)], *), \lambda m.\lambda m'.([\,], *))\ (succ\ k)) \\
&\equiv \pi_L((\lambda m.\lambda m'.([\,], *))\ k\ natrec(([\exists n^{\mathbb{N}}.(0 = succ\ n)], *), \lambda m.\lambda m'.([\,], *))\ k) \\
&\equiv \pi_L([\,], *) \\
&\equiv [\,]
\end{aligned}$$

Notions of Constraint

Central to the idea of constraint extraction is a *notion C of constraint*, a set including a *unit constraint* 1, together with a function $under\colon \mathcal{B} \times C \to \mathcal{B}$ satisfying the following conditions:

- $\phi\ under\ 1$ is always equivalent to ϕ
- constraints can be combined, so that for any constraints c, d there is a constraint $c.d$ such that $\phi\ under\ c\ under\ d$ is always equivalent to $\phi\ under\ c.d$.
- the application of constraints preserves implication: if ϕ implies ψ, then $\phi\ under\ c$ implies $\psi\ under\ c$, for every constraint c.

General constraints

The constraints relative to which $\bigcirc$ is interpreted form a monoidal action $\mathbf{C} \equiv (\mathbf{C}, 1, ., under)$ which preserves implication. Since different choices of $\mathbf{C}$ lead to different notions of logical refinement and constraint extraction, Mendler's original formulation of lax logic is rather general. In the *standard* interpretation $\mathbf{C}$ is the sets of lists $[c_1, \ldots, c_n]$ with members from some subset of statements, constraint composition is list concatenation, the void constraint is the empty list, and $under$ is:

$$\phi \; under \; [c_1, \ldots, c_n] \quad \equiv \quad c_n \rightarrow \cdots \rightarrow c_1 \rightarrow \phi$$

We can replace the constraint list $C = [c_1, \ldots, c_k]$ of the standard interpretation by the single constraint $\sqcap C$ where $\sqcap =_{\mathrm{def}} fold_{z,x}(true, z \wedge x)$.

Multiple constraint levels

The idea behind multi-level lax logic (MLL) is to allow *multiple* notions of constraint to operate simultaneously. Currently, all constraints must belong to the same underlying monoid.

We could instead use product notions, for example, but no general-purpose composition of constraint notions has been investigated, but it is reasonable to expect the cardinality $|\mathbf{C} \times \mathbf{C}'|$ to be of the order of $|\mathbf{C}| \times |\mathbf{C}'|$ or (even infinitely) worse. Consequently, if we attempt to solve systems defined relative to multiple notions of constraint, we are likely to run into combinatorial explosion problems and a consequent lack of scalability.

Operational MLL [LK06]

Write $p \triangleleft M$ to mean that p is a proof(-term) for the statement M. Ed's work considers expressions of the form

$$p_1 \triangleleft p_2 \triangleleft \ldots \triangleleft p_n \triangleleft \phi$$

and shows how to extract constraints at each level. These constraints satisfy statements of the form *this constraint allows us to deduce that that constraint allows us to deduce that the next constraint . . . allows us to deduce ϕ.* His approach is 'operational' in the following sense: he defines multi-level versions of the logical connectives and deduction rules, and then extends the translation rules given above for 'level-one' lax logic.

Example: $\bigcirc_n$

Having defined the operators `let` and `val`, Lewis defines deduction rules for the $\bigcirc$ operator.

$$\frac{\Gamma \vdash p_1 \lhd \ldots \lhd p_n \lhd P}{\Gamma \vdash \mathtt{val}_{n,1} p_1 \lhd \ldots \lhd \mathtt{val}_{n,n} p_n \lhd \bigcirc_n P} \bigcirc_n I$$

$$\frac{\Gamma \vdash p_1 \lhd \ldots \lhd p_n \lhd \bigcirc_n P \qquad \Gamma, z_1 \lhd \ldots \lhd z_n \lhd P \vdash q_1 \lhd \ldots \lhd q_n \lhd \bigcirc_n Q}{\begin{array}{c}\Gamma \vdash \mathtt{let}_{n,1} z_n \ldots z_1 \Leftarrow p_n \ldots p_1 \ \mathtt{in}\ q_n \ldots q_1 \lhd \ldots \lhd \\ \mathtt{let}_{n,n-1} z_n z_{n-1} \Leftarrow p_n p_{n-1} \ \mathtt{in}\ q_n q_{n-1} \lhd \\ \mathtt{let}_{n,n} z_n \Leftarrow p_n \ \mathtt{in}\ q_n \lhd \bigcirc_n Q\end{array}} \bigcirc_n E$$

Consolidation

Because Ed's rules are defined one connective at a time, he cannot guarantee *a priori* that his logic makes sense *as a whole*, but has to prove this. He has implemented the rules using both Lego and Isabelle, at the same time showing that his logic has 'sensible properties'. He has a translation T taking each $(n+1)$-level expression into an equivalent n-level expression. Ultimately, his approach appears to rely on the following claim (currently being checked):

Claim: Given any level n formula ϕ, we have $\vdash_{\mathcal{B}} T^n\phi$ if and only if $\vdash_n \phi$.

Recursive MLL

Another approach! Any suitably rich base logic $\mathcal{B}$ can be extended to a lax logic $\mathcal{L}$: write Ψ for this (essentially algorithmic) procedure. Build a transfinite *lax hierarchy* by defining (for ordinals ν and limit ordinals μ)

$$\begin{array}{rcl} \mathcal{L}_0 & =_{\mathrm{def}} & \mathcal{B} \\ \mathcal{L}_1 & =_{\mathrm{def}} & \mathcal{L} \\ \mathcal{L}_{\nu+1} & =_{\mathrm{def}} & \Psi\,\mathcal{L}_\nu \\ \mathcal{L}_\mu & =_{\mathrm{def}} & \bigcup\{\mathcal{L}_\nu \mid \nu < \mu\} \end{array}$$

Taking MLL to be $\bigcup_{n<\omega}\mathcal{L}_n$ gives the multi-level logic we seek. Easy result: If $\mathcal{B}$ is consistent, so is $\bigcup_{n<\omega}\mathcal{L}_n$. Note. Each laxification can be w.r.t. a different notion of constraint — the MLL type system may depend upon the choices made at each level.

Advice, please...

- Which approach to MLL makes more sense?
- Should the two approaches give equivalent logics?
- Is there any role for a transfinite version of MLL? What might we use it for?
- What about the `lets-not-bother` rule? How come a semantically empty rule is actually useful?!

Further Reading

[Acz99] P. Aczel. The Russell-Prawitz Modality. *Math. Struct. in Comp. Science*, 1999.

[Cur52] H.B. Curry. The Elimination Theorem when Modality is Present. *J. Symbolic Logic*, 17(4):249–265, 1952.

[FM97] M. Fairtlough and M. Mendler. Propositional Lax Logic. *Information and Computation*, 137(1):1–33, 1997.

[FW97] M. Fairtlough and M. Walton. Quantified Lax Logic. Technical Report CS–97–11, University of Sheffield, Department of Computer Science, 1997.

[LK06] E. Lewis-Kelham. *Multi-level Lax Logic*. PhD thesis, University of Sheffield, Department of Computer Science, 2006. Submittted.

[Men93] M. Mendler. *A Modal Logic for Handling Behavioural Constraints in Formal Hardware Verification*. PhD thesis, Edinburgh University, Department of Computer Science, 1993.

[PD99] F. Pfenning and R. Davies. A Judgemental Reconstruction of Modal Logic. Technical report, Carnegie-Mellon University, Department of Computer Science, 1999.

[Wal99] M. Walton. *First-Order Lax Logic: A Framework for Abstraction Constraints and Refinement,*. PhD thesis, 1999.

Thank you!

- Which approach to MLL makes more sense?
- Should the two approaches give equivalent logics?
- Is there any role for a transfinite version of MLL? What might we use it for?
- What about the `lets-not-bother` rule? How come a semantically empty rule is actually useful?!
- Any other questions worth addressing as well (or instead)?

Please contact me at: `M.Stannett@dcs.shef.ac.uk`