$\boxed{\Omega \vdash a :^{\ell} A}$ *(Typing)*

$$\text{T-Type}\ \frac{\mathbf{axiom}\ s_1\ s_2}{\Omega \vdash s_1 :^{\ell} s_2} \qquad \text{T-Conv}\ \frac{\Omega \vdash a :^{\ell} A \quad |\top \downarrow \Omega| \vdash A \equiv_{\ell_0} B \quad \top \downarrow \Omega \vdash B :^{\ell_0} s}{\Omega \vdash a :^{\ell} B} \qquad \text{T-Var}\ \frac{\ell_0 \leq \ell \qquad x :^{\ell_0} A \in \Omega}{\Omega \vdash x :^{\ell} A}$$

$$\text{T-Pi}\ \frac{\Omega \vdash A :^{\ell} s_1 \quad \Omega, x :^{\ell} A \vdash B :^{\ell} s_2 \quad \mathbf{rule\_pi}\ s_1\ s_2\ s_3}{\Omega \vdash \Pi x :^{\ell_0} A.B :^{\ell} s_3} \qquad \text{T-Abs}\ \frac{\Omega, x :^{\ell_0} A \vdash b :^{\ell} B \quad \top \downarrow \Omega \vdash (\Pi x :^{\ell_0} A.B) :^{\ell_1} s}{\Omega \vdash \lambda^{\ell_0} x : A.b :^{\ell} \Pi x :^{\ell_0} A.B}$$

$$\text{T-App}\ \frac{\Omega \vdash b :^{\ell} \Pi x :^{\ell_0} A.B \quad \ell \downarrow \Omega \vdash a :^{\ell_0} A}{\Omega \vdash b\ a^{\ell_0} :^{\ell} B\{a/x\}}$$

$$\text{T-WSigma}\ \frac{\Omega \vdash A :^{\ell} s_1 \quad \Omega, x :^{\ell} A \vdash B :^{\ell} s_2 \quad \mathbf{rule\_sig}\ s_1\ s_2\ s_3}{\Omega \vdash \Sigma x :^{\ell_0} A.B :^{\ell} s_3} \qquad \text{T-WPair}\ \frac{\top \downarrow \Omega \vdash (\Sigma x :^{\ell_0} A.B) :^{C} s \quad \ell \downarrow \Omega \vdash a :^{\ell_0} A \quad \Omega \vdash b :^{\ell} B\{a/x\}}{\Omega \vdash (a^{\ell_0}, b) :^{\ell} \Sigma x :^{\ell_0} A.B}$$

$$\text{T-LetPair}\ \frac{\top \downarrow \Omega \vdash (\Pi y :^{\ell} B.C) :^{\ell_1} s \quad \Omega \vdash a :^{\ell} (\Sigma x :^{\ell_0} A.B) \quad \Omega, x :^{\ell_0} A \vdash c :^{\ell} (\Pi y :^{k} B.C)}{\Omega \vdash \mathbf{let}\ (x^{\ell_0},) = a\ \mathbf{in}\ c :^{\ell} C}$$

$$\begin{aligned} \ell \downarrow \varnothing &= \varnothing \\ \ell \downarrow \Omega, x :^{\ell_0} A &= \ell \downarrow \Omega, x :^{\ell_0} A \text{ if } \neg(\ell_0 \leq \ell) \\ \ell \downarrow \Omega, x :^{\ell_0} A &= \ell \downarrow \Omega, x :^{\bot} A \text{ if } \ell_0 \leq \ell \end{aligned}$$

Figure 1: Typing

Some high-level ideas on why the system is cool:

- It unifies the two relevance tracking mechanisms in DDC into a single one. There is no special case for when ℓ is top. In fact, the system does not even require a top element for the labels.
- The system is expressive enough to encode strong existentials. I believe all the interesting examples that DDC supports are also supported in this reformulated system.
- During compile-time reasoning, this system can ignore more terms than

DDC. (todo: write down those examples) The expressiveness stems from the fact that the system squashes the observer label more often than DDC. (will explain what squash means in this system later)

- ...

Some notable changes compared to DDC:

- The labels ℓ must form a linear order. This is necessary for consistency to hold. It is okay for the linear order to be top-less.
- The typing judgment can take any label ℓ. The base cases such as rule T-TYPE, rule T-VAR no longer exclude the top element (in this reformulated version, l may not even have a top element).
- Well-formedness for types are no longer checked at the top level. Instead, we consider a type to be well-formed as long as there exists some label ℓ_1 where we can check the type. This design can be seen in both rule T-ABS and rule T-CONV (the notation $\ell \downarrow \Omega$ will be explained in the next bullet point).
- The system introduces the squash operation $\ell \downarrow \Omega$ (bottom of Figure 1), which serves a similar function to resurrection but works slightly differently. Given a label ℓ, $\ell \downarrow \Omega$ returns a context with the labels visible at ℓ squashed to bottom and everything else left unchanged. One use case of the squash operation can be found in rule T-APP. When applying a function b to a term a boxed at level ℓ_0, we squash Ω with the current label ℓ before checking a at level ℓ_0. This subsumes the two application rules from DDC. The squashing on the context is necessary so we can allow a relevant function to be applied to an irrelevant argument that appears in an irrelevant position. DDC implements this idea by joining ℓ and ℓ_0 before checking a. DDC's implementation loses some opportunities for compile-time irrelevance from raising the label. (need a few examples to explain this better)
- Rule T-CONV shows how compile-time irrelevance is implemented in the system. The equality between A and B is graded at some level ℓ_0 rather than a specific top or run-time irrelevant level. The intuition is as follows: If A is well-graded at ℓ_0, then we are licensed to use any equalities that are at level ℓ_0 or above. This is definitely consistent. I can explain why that's the case in person. It relies on a specific property about parallel reduction that I've already proven in Coq (it's formulated in DE, but should be portable to DDC).
- The equality rules are omitted, though they do need to be revised accordingly based on the the typing judgment presented here. I want to refactor the equality rules so they are typed. I find the well-graded judgment and graded equality much harder to work with than a self-contained typed equality.