Ordinal Collapsing Functions and Taranovsky Ordinal Notation

Jacques Bailhache (jacques.bailhache@gmail.com)

July 31, 2021

1 Ordinal collapsing functions and Taranovsky Ordinal Notation

One way to define large countable ordinals is to collapse a large (uncountable) cardinal with an ordinal collapsing function (OCF).
In http://www.madore.org/ david/math/ordinal-zoo.pdf David Madore enumerates some examples of ordinals defined this way, as the countable collapse of $epsilon_{\kappa+1}$ (which is the limit of $\kappa, \kappa^{\kappa}, \kappa^{\kappa^{\kappa}}, ...$) where κ is respectively :

- the first inaccessible cardinal I
- the first Mahlo cardinal M
- the first weakly compact cardinal K
- the first Π^0_2 indescribable cardinal

The definition of large cardinals is very complex, as shown by David Madore's blog post
http://www.madore.org/ david/weblog/d.2017-08-31.2462.ordinaux-interessants.html :

> Pour aller encore plus loin, on fait appel à un cardinal « faiblement compact » K, une notion de grand cardinal (domaine (3)) dont l'analogue récursif (domaine (2)) s'appelle « Π_3-réfléchissant », quelque chose que je ne compte pas définir, mais à ce niveau-là, le système de fonctions d'écrasement devient encore plus complexe. Au niveau de l'écrasement d'un cardinal récursivement Mahlo (:= Π_2-réfléchissant sur les admissibles, c'est-à-dire Π_2-réfléchissant sur les Π_2-réfléchissants), que j'avais évoqué ci-dessus, il y avait deux « niveaux » de fonctions d'écrasement : θ qui sert à fabriquer des cardinaux réguliers / ordinaux admissibles π<M, et toutes sortes de fonctions $\psi\pi$ qui fabriquent des ordinaux $<\pi$. Maintenant qu'on veut jouer avec K, les choses se compliquent encore : on a une hiérarchie par niveau de « Mahloïtude », avec au niveau 0 tous les ordinaux <K, au niveau 1 les cardinaux réguliers ou ordinaux admissibles, au niveau 2 les cardinaux Mahlo ou ordinaux récursivement Mahlo, et au niveau γ+1 les cardinaux réguliers dans lesquels ceux du niveau γ sont clos-cofinaux (ou quelque chose d'analogue côté récursif : les ordinaux Π_2-réfléchissant sur ceux de niveau γ) ; on a une fonction d'écrasement qui produit le plus petit élément de chaque niveau de Mahloïtude, mais surtout, on a en a une autre $\psi\xi\pi$ de trois variables qui prend un niveau de Mahloïtude ξ, un π dans lequel écraser (qui est lui-même d'un certain niveau de Mahloïtude $>\xi$), et un ordinal à écraser, et qui produit un écrasement $<\pi$ de niveau de Mahloïtude ξ (qui peut donc servir pour de nouveaux écrasements si ξ>0). Bref, là où au paragraphe précédent on avait juste le niveau 0 (la fonction ψ) et une ébauche de niveau 1 (la fonction θ), on a maintenant toute une hiérarchie de niveaux, et K peut servir à diagonaliser sur la hiérarchie elle-même (produire le plus petit cardinal dont le niveau de Mahloïtude est égal à lui-même, etc.).
> L'ordinal $\psi 0\Omega(\epsilon K+1)$ décrit par tout ce fatras mesure la force d'une théorie qui est la théorie des ensembles de Kripke-Platek enrichie d'un axiome de Π_3-réflexion.
> Tout ça est déjà bien compliqué. Chronologiquement, cette analyse date de 1994 (Michael Rathjen, Proof theory of reflection, Ann. Pure Appl. Logic 68 (1994), 181–224 ; disponible ici mais avec malheureusement plein de fautes typographiques pénibles). C'est l'ordinal calculable le plus grand que j'ai l'impression de bien comprendre, au sens où je crois pouvoir coder sur un ordinateur un système de notations ordinales jusqu'à ce $\psi 0\Omega(\epsilon K+1)$ (ce n'est pas un exploit : l'article de Rathjen donne en gros tous les algorithmes nécessaires).
> Mais au-delà de ça, il arrive des complications bien plus importantes : pour écraser un ordinal « Π_4-réfléchissant », on doit commencer à gérer des ordinaux dont la description est vraiment plus complexe que l'écrasement de quelque chose (par exemple des ordinaux Π_2-réfléchissant sur les Π_3-réfléchissants) : les fonctions d'écrasement prennent en argument non pas juste un ordinal vers lequel écraser et un simple niveau de Mahloïtude, mais des données beaucoup plus riches que sont des « configurations de réflexion » ou des « instances de réflexion » (on

n'écrase pas juste vers un ordinal de niveau de Mahloïtude ξ et inférieur à π mais vers un ordinal ayant certaines propriétés de réflexion qui conduisent elles-mêmes à d'autres fonctions d'écrasement), et le système de notation devient incroyablement plus subtil et défini par un nombre assez impressionnant de récursions imbriquées. Au moins les ordinaux « Π_5-réfléchissants » ou plus n'apportent-ils pas plus de complexité substantielle par rapport aux Π_4-réfléchissants, mais il y a encore quelques subtilités si on veut inclure tous les niveaux d'un coup, voire, des niveaux indicés par le système d'ordinaux qu'on est en train de définir. C'est en gros à ce point-là que travaille la thèse de Jan-Carl Stegert (Ordinal proof theory of Kripke-Platek set theory augmented by strong reflection principles (2010), disponible ici en PDF), qui introduit des systèmes de notations ordinales dont la seule définition s'étend sur un bon nombre de pages (notamment p. 13–30 pour le système principal, p. 68–70 pour une version simplifiée, p. 66–67 pour une version encore plus simplifiée équivalente à l'écrasement d'un cardinal Mahlo / ordinal Π_3-réfléchissant, et p. 100–113 pour un système encore plus riche). De ce que je sais, c'est le système de notations ordinales explicite le plus grand qui ait été introduit et rigoureusement analysé dans la littérature mathématique.

Another (conjectured) way to define large countable ordinals (and also large (uncountable) cardinals) is Taranovsky Ordinal Notation (TON). This formalism is much more simpler than large cardinals.
Correspondences between TON and OCF can be found in :

- https://googology.wikia.org/wiki/User_blog:Boboris02/Analysis_of_Taranovsky
- https://googology.wikia.org/wiki/User_blog:Boboris02/Algorithm_for_Generating_LUCOs_From_TON_Expressions_
- https://googology.wikia.org/wiki/User_blog:Hyp_cos/TON,_stable_ordinals,_and_my_array_notation

In https://googology.wikia.org/wiki/User_blog:Boboris02/Algorithm_for_Generating_LUCOs_From_TON_Expressions_
the following notations are introduced :
$a = C(\Omega_2, 2, 0)$
$\pi_+ = C(\Omega_2, \pi)$
In https://googology.wikia.org/wiki/User_blog:Hyp_cos/TON,_stable_ordinals,_and_my_array_notation
an ordinal d is defined by :
$d = C(\Omega_2, C(\Omega_2 2, 0))$
So we have $d = a_+$.
TON standard form uses only 0, C and Ω_n, not arithmetic operators +, product... For example $\Omega_2 2 = C(\Omega_2, \Omega_2)$.
Check it with Taranovsky's ordinal arithmetic program :

```
pi@raspberrypi:/perso/ord $ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> execfile("oa.py")
>>> C2(W(2),W(2))
W(2)*2
>>>
```

$C(a,b) = b + \omega^a if C(a,b) \geq a$ We have $d = \omega^d, d^2 = \omega^d \omega^d = \omega^{d2}; d2 = d+d = \omega^d + d = C(d,d); d^2 = C(C(d,d),d) (not C(C(d,d),0) because d); d^d = C(C(C(d,d),d),d); d^{d^d} = C(C(C(C(d,d),d),d),d); ...$

```
pi@raspberrypi:/perso/ord $ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> execfile("oa.py")

>>> d=C2(W(2),C2(W(2)*2,0))
>>> d
C(W(2)**2+W(2))

>>> d**2
C(W(2)**2+W(2))**2
```

```
>>> C2(C2(d,d),0)
C(C(W(2)**2+W(2))**2)

>>> C2(C2(d,d),d)
C(W(2)**2+W(2))**2


pi@raspberrypi:/perso/ord $ hugs ton_explore.hs
__   __ __  __  ____   ___      ------------------------------------------
||   || ||  || ||  || ||__      Hugs 98: Based on the Haskell 98 standard
||___|| ||__|| ||__||  __||     Copyright (c) 1994-2005
||---||         ___||           World Wide Web: http://haskell.org/hugs
||   ||                         Bugs: http://hackage.haskell.org/trac/hugs
||   || Version: September 2006 ------------------------------------------

Haskell 98 mode: Restart with command line option -98 to enable extensions

Type :? for help

Main> let d = C W (C (C W W) 0) in C (C d d) 0 > d
False
Main> let d = C W (C (C W W) 0) in C (C d d) d > d
True
```

According to https://googology.wikia.org/wiki/User_blog:Hyp_cos/TON,_stable_ordinals,_and_my_array_notation we have :

- $I = C(\Omega_2 + d, 0)$
- $M = C(\Omega_2 + d^2, 0)$
- $K = C(\Omega_2 + d^d, 0)$

The standard forms are :

- $\Omega_2 + d = \Omega_2 + \omega^d = C(d, \Omega_2)$
- $\Omega_2 + d^2 = \Omega_2 + \omega^{d2} = C(d2, \Omega_2) = C(C(d, d), \Omega_2)$
- $\Omega_2 + d^d = \Omega_2 + \omega^{d^2} = C(d^2, \Omega_2) = C(C(C(d, d), d), \Omega_2)$

```
pi@raspberrypi:/perso/ord $ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> execfile('oa.py')
>>> d = C2(W(2),C2(C2(W(2),W(2)),0))
>>> d
C(W(2)**2+W(2))
>>> W(2)+d
W(2)+C(W(2)**2+W(2))
>>> C2(d,W(2))
W(2)+C(W(2)**2+W(2))
>>> C2(d*2,W(2))
W(2)+C(W(2)**2+W(2))**2
>>> C2(C2(d,d),W(2))
W(2)+C(W(2)**2+W(2))**2
>>> W(2)+d**d
W(2)+C(W(2)**2+W(2))**C(W(2)**2+W(2))
```

```
>>> C2(d**2,W(2))
W(2)+C(W(2)**2+W(2))**C(W(2)**2+W(2))
>>> C2(C2(C2(d,d),d),W(2))
W(2)+C(W(2)**2+W(2))**C(W(2)**2+W(2))
>>>
```