From Chaotic Iteration to Constraint Propagation

Krzysztof R. Apt

CWI
P.O. Box 94079, 1009 AB Amsterdam, The Netherlands
and
Dept. of Mathematics, Computer Science, Physics & Astronomy
University of Amsterdam, The Netherlands

"Don't express your ideas too clearly. Most people think little of what they understand, and venerate what they do not."

(The Art of Worldly Wisdom, Baltasar Gracián, 1647.)

Abstract. We show how the constraint propagation process can be naturally explained by means of chaotic iteration.

1 Introduction

1.1 Motivation

Over the last ten years constraint programming emerged as an interesting and viable approach to programming. In this approach the programming process is limited to a generation of requirements ("constraints") and a solution of these requirements by means of general and domain specific methods. The techniques useful for finding solutions to sets of constraints were studied for some twenty years in the field of Constraint Satisfaction. One of the most important of them is constraint propagation, the elusive process or reducing a constraint satisfaction problem to another one that is equivalent but "simpler".

The algorithms that achieve such a reduction usually aim at reaching some "local consistency", which denotes some property approximating in some loose sense "global consistency", so the consistency of the whole constraint satisfaction problem. (In fact, most of the notions of local consistency are neither implied by nor imply global consistency.)

For some constraint satisfaction problems such an enforcement of local consistency is already sufficient for finding a solution or for determining that none exists. In some other cases this process substantially reduces the size of the search space which makes it possible to solve the original problem more efficiently by means of some search algorithm.

The aim of this paper is to show that the constraint propagation algorithms can be naturally explained by means of chaotic iteration, a basic technique used
for computing limits of iterations of finite sets of functions that originated from
numerical analysis (see Chazan and Miranker (1969)) and was adapted for com-
puter science needs by Cousot and Cousot (1977). In fact, several constraint
propagation algorithms proposed in the literature turn out to be instances of
generic chaotic iteration algorithms studied here.

Moreover, by characterizing a given notion of a local consistency as a common
fixed point of a finite set of monotonic and inflationary functions we can auto-
matically generate an algorithm achieving this notion of consistency by “feeding”
these functions into a generic chaotic iteration algorithm.

1.2 Preliminaries

Definition 1. Consider a sequence of domains \( D := D_1, \ldots, D_n \).

- By a scheme (on \( n \)) we mean a sequence of different elements from \([1..n]\).
- We say that \( C \) is a constraint (on \( D \)) with scheme \( i_1, \ldots, i_l \) if \( C \subseteq D_{i_1} \times \cdots \times D_{i_l} \).
- Let \( s := s_1, \ldots, s_k \) be a sequence of schemes. We say that a sequence of
  constraints \( C_1, \ldots, C_k \) on \( D \) is an \( s \)-sequence if each \( C_i \) is with scheme \( s_i \).
- By a Constraint Satisfaction Problem \( \langle D;C \rangle \), in short CSP, we mean a se-
  quence of domains \( D \) together with an \( s \)-sequence of constraints \( C \) on \( D \). We
call then \( s \) the scheme of \( \langle D;C \rangle \).

Given an \( n \)-tuple \( d := d_1, \ldots, d_n \) in \( D_1 \times \cdots \times D_n \) and a scheme \( s := i_1, \ldots, i_l \) on \( n \) we denote by \( d[s] \) the tuple \( d_{i_1}, \ldots, d_{i_l} \). In particular, for \( j \in [1..n] \) \( d[j] \) is
the \( j \)-th element of \( d \). By a solution to a CSP \( \langle D;C \rangle \), where \( D := D_1, \ldots, D_n \),
we mean an \( n \)-tuple \( d \in D_1 \times \cdots \times D_n \) such that for each constraint \( C \) in \( C \)
with scheme \( s \) we have \( d[s] \in C \).

Consider now a sequence of schemes \( s_1, \ldots, s_k \). By its union, written as
\( \langle s_1, \ldots, s_k \rangle \) we mean the scheme obtained from the sequences \( s_1, \ldots, s_k \) by re-
moving from each \( s_i \) the elements present in some \( s_j \), where \( j < i \), and by con-
catenating the resulting sequences. For example, \( \langle (3,7,2), (4,3,7,5), (3,5,8) \rangle =
(3,7,2,4,5,8) \). Recall that for an \( s_1, \ldots, s_k \)-sequence of constraints \( C_1, \ldots, C_k \) their
join, written as \( C_1 \Join \cdots \Join C_k \), is defined as the constraint with scheme
\( \langle s_1, \ldots, s_k \rangle \) and such that

\[ d \in C_1 \Join \cdots \Join C_k \text{ iff } d[s_i] \in C_i \text{ for } i \in [1..k]. \]

Further, given a constraint \( C \) and a subsequence \( s \) of its scheme, we denote
by \( \Pi_s(C) \) the constraint with scheme \( s \) defined by

\[ \Pi_s(C) := \{ d[s] \mid d \in C \}, \]

and call it the projection of \( C \) on \( s \). In particular, for a constraint \( C \) with scheme
\( s \) and an element \( j \) of \( s \), \( \Pi_j(C) = \{ a \mid \exists d \in C \ a = d[j] \} \).

Given a CSP \( \langle D;C \rangle \) we denote by \( \text{Sol}(\langle D;C \rangle) \) the set of all solutions to it.
If the domains are clear from the context we drop the reference to \( D \) and just
write \( \text{Sol}(C) \). The following observation is useful.